UNIVERSITÀ DEGLI STUDI DI UDINE

DIPARTIMENTO POLITECNICO DI INGEGNERIA E ARCHITETTURA

PH.D IN INDUSTRIAL AND INFORMATION ENGINEERING

Synchronization Problems in Computer Vision

Author:
Federica ARRIGONI

Supervisor:
Prof. Andrea FUSIELLO

October 31, 2017

Abstract

The goal of "synchronization" is to infer the unknown states of a network of nodes, where only the ratio (or difference) between pairs of states can be measured. Typically, states are represented by elements of a group, such as the Symmetric Group or the Special Euclidean Group. The former can represent local labels of a set of features, which refer to the multi-view matching application, whereas the latter can represent camera reference frames, in which case we are in the context of structure from motion, or local coordinates where 3D points are represented, in which case we are dealing with multiple point-set registration. A related problem is that of "bearing-based network localization" where each node is located at a fixed (unknown) position in 3-space and pairs of nodes can measure the direction of the line joining their locations. In this thesis we are interested in global techniques where all the measures are considered at once, as opposed to incremental approaches that grow a solution by adding pieces iteratively.

To my Parents…

Contents

Chapter 1

Introduction

Consider a network of nodes where each node is characterized by an unknown state, and suppose that pairs of nodes can measure the ratio (or difference) between their states. The goal of *synchronization* [98, 167] is to estimate the unknown states from the pairwise measures. The problem can be profitably modeled as a graph where nodes correspond to the unknown states and edges encode the pairwise measures, and it is well-posed only if such a graph is connected. Solving a synchronization problem can be seen as upgrading from relative (pairwise) information, which involves two nodes at a time, to absolute (global) information, which involves all the nodes simultaneously. It can be shown that this is equivalent to enforce *cycle consistency* [61, 198], namely the property that the composition of relative measures along any cycle in the graph should return the identity element.

Mathematically, states are represented by elements of a group Σ. According to the chosen group, we have several instances of synchronization: $\Sigma = \mathbb{R}$ yields *time synchronization* [98, 75], from which the term *synchronization* originates; $\Sigma = \mathbb{R}^d$ is a *translation synchronization* [13, 154, 179, 132, 3]; $\Sigma = SO(d)$ corresponds to *rotation synchronization* (also known as rotation averaging) [163, 126, 167, 52, 70, 28, 29, 86, 44, 187, 7, 182] and $\Sigma = SE(d)$ results in *rigid-motion synchronization* (also known as motion averaging or pose-graph optimization) [71, 79, 177, 180, 20, 10, 9, 151, 150]; finally, $\Sigma = Sym(d)$ gives rise to *permutation synchronization* [143, 164, 197, 8]. In this thesis we will deal with $\Sigma = SE(d)$ and $\Sigma = Sym(d)$.

In the case of permutation synchronization, each state is an unknown reordering of d objects, which is represented as a $d \times d$ permutation matrix, namely an element of the Symmetric Group $Sym(d)$. It is assumed that pairs of nodes can match these objects, establishing which objects are the same in the two nodes, despite the different naming, and the goal is to infer a global labeling of the objects, such that the same object receives the same label in all the nodes. Permutation synchronization finds application in *multi-view matching* [47, 205], where nodes are images and objects are features. In practice, not all the features are visible in all the images, so matches are modeled as *partial* permutations, which form the so-called Symmetric Inverse Semigroup $ISym(d)$. In this thesis we build upon [167] and develop a novel solution to partial permutation synchronization based on a spectral decomposition, which successfully handles missing correspondences.

In the case of rigid-motion synchronization, each state is the angular attitude and

position of a d-dimensional reference frame, which is referred to as *motion* in Computer Vision, *orientation* in Photogrammetry, or *pose* in Robotics. Mathematically, each motion/orientation/pose is described by a direct isometry, which is an element of the Special Euclidean Group $SE(d)$. Therefore the goal is to recover the location and attitude of a set of reference frames organized in a network, where the links of this network are relative transformations of one frame with respect to the others. If we restrict the attention to the angular attitude (leaving out the position) then we get rotation synchronization. Similarly, if position only is considered, it results in translation synchronization. Such local frames can be camera reference frames, in which case we are in the context of *structure-from-motion* [142], or local coordinates where 3D points are represented, in which case we are dealing with *multiple point-set registration* [147]. In the first case, the goal is to recover both the 3D structure of the scene and camera motion from multiple images, whereas in the second case the goal is to find the rigid transformations that bring multiple 3D point sets into alignment.

In this thesis we express synchronization over $SO(d)$ and $SE(d)$ in terms of *low-rank and sparse* (LRS) matrix decomposition, that is the problem of recovering a low-rank matrix starting from an incomplete subset of its entries, possibly corrupted by noise and outliers. As far as the Special Euclidean Group is concerned, we tackle the problem of multiple point-set registration, whereas in the case of the Special Orthogonal Group we concentrate on structure from motion. We also propose a closed-form approach to synchronization over $SE(d)$ which is applied to multiple point-set registration, where robustness to outliers is gained via iteratively re-weighted least squares (IRLS). This method can be viewed as the extension to $SE(d)$ of the spectral solution developed in [167, 4, 168] for $SO(d)$.

Note that the motion task of the structure-from-motion problem cannot be straightforwardly solved as a synchronization over $SE(3)$, due to the depth-speed ambiguity. Indeed, only the *directions* of the relative displacements between camera pairs can be measured, but the magnitude is unknown. A possibility consists in breaking the motion estimation process in two stages, namely a rotation synchronization to obtain the angular attitudes of the cameras, followed by the recovery of camera positions from pairwise directions, which is an instance of *bearing-based network localization* [201] in 3-space where sensors are the cameras. This workflow is exploited in several structure from motion systems, such as [78, 33, 96, 195, 76]. Alternatively, the translation magnitudes (referred to as the *epipolar scales*) can be explicitly computed, as we propose in this thesis via a two-stage method: first, a *cycle basis* is computed; then, all the epipolar scales are recovered simultaneously by solving a homogeneous linear system. This allows either to address the synchronization problem over $SE(3)$, using (e.g.) the spectral solution, or to perform rotation synchronization followed by translation synchronization. With reference to the latter, we also propose a "divide and conquer" technique for computing the epipolar scales.

The two paths are equivalent. In fact, we show that the epipolar scales can be

uniquely (up to scale) recovered from pairwise directions if and only if node locations can be uniquely (up to translation and scale) recovered from pairwise directions. Requiring that the graph is connected is not sufficient for unique localizability, but more complicated assumptions are required, which are studied under the name of *parallel rigidity* [192, 63]. Several theoretical results about parallel rigidity are present in the literature, which come from disparate communities, including discrete geometry, computer vision, and robotics. In this thesis we provide a unifying view of the problem, rewriting and linking in a coherent structure several results that were presented in different contexts with disparate formalisms.

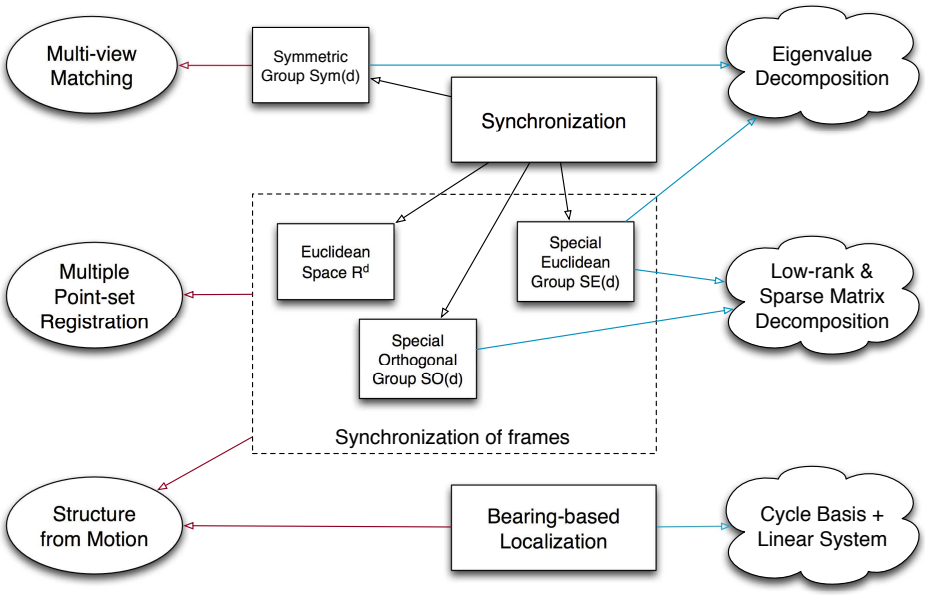


FIGURE 1.1: Conceptual diagram of the thesis. Rectangles refer to the theoretical problems we analyzed, cloud-like shapes show the solutions we proposed to address these problems, and circles denote the applications we considered in the experimental evaluation.

The thesis, whose conceptual diagram is drawn in Figure 1.1, is organized as follows. Chapter 2 is devoted to the synchronization problem, which is studied from a theoretical perspective, and practical algorithms are also proposed to solve it. Chapter 3 considers the bearing-based localization problem, with particular focus to the localizability aspect. Some applications of synchronization are detailed in Chapter 4, with particular attention to structure from motion, which constitutes the connection between synchronization and bearing-based localization. The solutions proposed in this thesis are supported by experimental results on synthetic data, which are reported in Chapter 5, and on real data, which are shown in Chapter 6, within the context of multi-view matching, multiple point-set registration and structure from motion. Conclusion and possible future work are drawn in Chapter 7. The results presented in this thesis require the definitions of the Kronecker, Khatri-Rao and Hadamard products, which are given in Appendix A, and some basic notions from graph theory, which are covered in Appendix B. An introduction to low-rank and sparse matrix decomposition is presented in Appendix C.

Part of the material contained in this thesis is taken from the following papers.

1. Federica Arrigoni and Andrea Fusiello. Synchronization problems in computer vision. *Submitted*.

2. Federica Arrigoni and Andrea Fusiello. Bearing-based network localizability: a unifying view. *Submitted*.

3. Federica Arrigoni, Beatrice Rossi, Pasqualina Fragneto and Andrea Fusiello. Robust synchronization in SO(3) and SE(3) via low-rank and sparse matrix decomposition. *Submitted*.

4. Federica Arrigoni, Eleonora Maset and Andrea Fusiello. Synchronization in the symmetric inverse semigroup. *International Conference on Image Analysis and Processing (ICIAP)*, volume 10485 of *Lecture Notes in Computer Science*, pages 70 – 81. Springer International Publishing, 2017.

5. Eleonora Maset, Federica Arrigoni and Andrea Fusiello. Practical and efficient multi-view matching. *International Conference on Computer Vision (ICCV)*, pages 4568 – 4576, 2017.

6. Federica Arrigoni, Beatrice Rossi and Andrea Fusiello. Spectral synchronization of multiple views in SE(3). *SIAM Journal on Imaging Sciences*, 9(4):1963 – 1990, 2016.

7. Federica Arrigoni, Andrea Fusiello and Beatrice Rossi. Camera motion from group synchronization. *International Conference on 3D Vision (3DV)*, pages 546–555, 2016.

8. Federica Arrigoni, Beatrice Rossi and Andrea Fusiello. Global registration of 3D point sets via LRS decomposition. *European Conference on Computer Vision (ECCV)*, volume 9908 of *Lecture Notes in Computer Science*, pages 489–504. Springer International Publishing, 2016.

9. Federica Arrigoni, Andrea Fusiello and Beatrice Rossi. On computing the translations norm in the epipolar graph. *International Conference on 3D Vision (3DV)*, pages 300–308, 2015.

10. Federica Arrigoni, Beatrice Rossi and Andrea Fusiello. Robust and efficient camera motion synchronization via matrix decomposition. *International Conference on Image Analysis and Processing (ICIAP)*, volume 9279 of *Lecture Notes in Computer Science*, pages 444–455. Springer International Publishing, 2015.

Chapter 2

Synchronization

In this chapter we survey and put in a common framework several works that have been developed in different contexts, all dealing with the same abstract problem, called *synchronization* by some authors, or averaging, or graph optimization by others. The problem is formulated in a group and requires to find elements of the group given a certain number of their mutual differences (or ratios, depending on how the group operation is called). In particular, we concentrate on a matrix formulation of synchronization, which leads to a neat theory and closed-form solutions.

2.1 Introduction

Consider a network of nodes where each node is characterized by an unknown state, and suppose that pairs of nodes can measure the ratio (or difference) between their states. The goal is to infer the unknown states from the pairwise measures. This is a general statement of the *synchronization* problem [98, 75, 167]. States are represented by elements of a group Σ, that is why the problem is actually referred to as *group synchronization*.

The problem can be usefully modelled by introducing a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is referred to as the *measurement graph*, where nodes correspond to the unknown states and edges correspond to the pairwise measures, and it is well-posed only if such a graph is connected. In the literature, the same problem is also referred to as *averaging* [79, 194, 86] or *graph optimization* [41].

As an example, consider the graph reported in Figure 2.1, where nodes and edges are labelled with integer numbers: the task is to recover the unknown numbers in the nodes by measuring their differences (on the edges). Two things can be immediately observed: a solution exists only if the sum of the differences along any cycle is zero, and, when it exists, the solution is not unique, for adding a constant to the nodes does not change the differences.

Measures are typically corrupted by errors, which can be gross errors (outliers) and/or a diffuse noise with small variance. If $\mathcal{G}$ is a tree then these errors will creep in the solution, however, as soon as redundant measures are considered (i.e. the graph has at least one cycle), they are exploited by the synchronization process to globally compensate the errors.

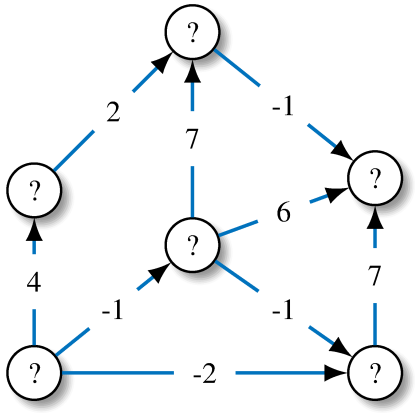


FIGURE 2.1: Synchronization over $(\mathbb{Z}, +)$.

The solution minimizes a suitable cost function which evaluates the consistency between the unknown states and the pairwise measures, which is equivalent to impose that the composition of relative measures along any cycle in the measurement graph should return the identity element (or be as close as possible, in the noisy case) [61, 198].

Several instances of synchronization have been studied in the literature. $\Sigma = \mathbb{R}$ yields *time synchronization* [98, 75], from which the term *synchronization* originates, where all the nodes in a network are synchronized to a common clock. $\Sigma = \mathbb{Z}_2$ gives *sign synchronization* [54], which was used to identify communities in a network where the interaction between the nodes is described by two dichotomous values, e.g., agreement/disagreement, and the network has a natural partition into two communities. $\Sigma = \mathbb{R}^d$ is a *translation synchronization* [13, 154, 179, 132, 3], namely the problem of localizing a set of nodes in space from pairwise differences. $\Sigma = SO(d)$ corresponds to *rotation synchronization* (also known as rotation averaging) [163, 126, 167, 52, 70, 28, 29, 86, 44, 187, 7, 182] and $\Sigma = SE(d)$ results in *rigid-motion synchronization* (also known as motion averaging or pose graph optimization) [71, 79, 177, 180, 20, 10, 9, 151, 150], which find application in structure from motion, registration of 3D point sets and simultaneous localization and mapping. $\Sigma = SL(d)$ produces *homography synchronization* [159], which is an essential step in the context of image stitching or mosaicking. Finally, $\Sigma = Sym(d)$ gives rise to *permutation synchronization* [143, 164, 197, 8], which is related to multi-view matching.

2.1.1 Related Work

While it is clear that time/translation synchronization can be reduced to a linear system of equations, the other instances of synchronization involve the minimization of a non convex cost function, which make the problem difficult to solve.

However, if Σ admits a matrix representation, i.e. it can be embedded in $\mathbb{R}^{d\times d}$, then the synchronization problem can be expressed as an eigenvalue decomposition,

resulting in an efficient and closed-form solution. Specifically, the unknown states are derived from the top eigenvectors of a matrix constructed from the pairwise measures. This procedure was introduced in [167] for $\Sigma = SO(2)$, extended in [4, 168] to $\Sigma = SO(3)$ and further generalized in [143, 164] to $\Sigma = Sym(d)$. The same formulation appeared in [159] for $\Sigma = SL(d)$. An equivalent null-space formulation can also be derived [126]. A closed-form solution to synchronization over $SO(3)$ is also developed in [78] where rotations are represented as unit quaternions and the problem is expressed as a linear system of equations.

A matrix representation of the group is also exploited in [79, 2, 82], with particular focus on the $\Sigma = SO(3)$ and $\Sigma = SE(3)$ cases, where an iterative scheme is formulated based on Lie group theory, in which at each step the unknown states are updated by averaging relative measures in the Lie algebra. As it will be shown in Section 2.5.4, this can be viewed as a translation synchronization problem. In [44, 45] robustness is added to the original technique through Iteratively Reweighted Least Squares (IRLS).

Other techniques exploiting a matrix representation of the group can be found in the literature, which express the synchronization problem in terms of well studied mathematical tools, such as semidefinite programming [167, 187, 151] or matrix completion [11], resulting in iterative solutions. Note that the semidefinite programming formulation derives from specific properties of $O(d)$ and $SE(d)$, namely the fact that the matrix constructed from the pairwise measures is positive semidefinite when considering the Orthogonal Group [167, 187], and the property that the convex hull of the Special Euclidean Group admits a semidefinite representation [151]. A convex relaxation is also employed in [150] where the authors, using the theory of Lagrangian duality, develop an algorithm for certifying the global optimality of a candidate solution to synchronization over $SE(d)$.

Other approaches, which are briefly reviewed here, include iterating local solutions on the measurement graph [163, 59, 145, 85, 1, 177] or explicit minimization of a cost function [197, 42, 70, 52, 180, 181, 28].

The authors of [163] decompose the measurement graph into a set of cycles, and they propose an iterative procedure to recover the unknown states in which the error is distributed over these cycles. The same idea appeared also in [59, 145]. In [85] a cost function based on the ℓ_1-norm is minimized to solve rotation synchronization, where each state is updated in turn by applying the Weiszfeld algorithm to its neighbours. This technique is generalized to ℓ_q-optimization in [1], with $1 \leq q < 2$, where improved reliability and robustness is shown compared to using the ℓ_2-norm. A similar approach is adopted in [177] for rigid-motion synchronization, where each state is updated in turn in a distributed fashion, exploiting the fact that rigid motions can be represented by dual quaternions, in the same way as rotations can be represented by quaternions.

In [197] the permutation synchronization problem is addressed via the Gauss-Seidel method, where the set $Sym(d)$ is relaxed to its convex hull. In [42] Quasi-Newton iterations are used whereas in [70] the theory of Lagrangian duality is exploited for synchronization over $SE(3)$ and $SO(3)$, respectively. Both [42] and [70] use unit quaternions to parametrize rotations. However, according to the analysis in [126], methods involving matrices usually perform better than those that use quaternions. The authors of [52] use a truncated quadratic as a more robust cost function for rigid-motion synchronization, which is minimized via a discrete Markov random field formulation, combined with a continuous Levenberg-Marquardt refinement. In addition to relative measures, vanishing points and information from other sensors are assumed as input, with reference to the structure from motion application. However, these are local iterative methods, hence they require a good initialization.

The authors of [180, 181] exploit the Riemannian manifold structure of $SO(3)$ and $SE(3)$ and compute the unknown states via Riemannian gradient descent. In [180] non isotropic noise and incomplete measurements are taken into account through the use of covariance matrices, whereas in [181] the choice of the step-size and several cost functions are discussed, including a reshaped cost function, which is similar to robust M-estimators and is less sensitive to large errors. In [28] a statistical approach is adopted by assuming a specific noise model that takes into account also the presence of outliers, and a maximum likelihood estimator is computed via Riemannian trust-region optimization.

2.1.2 Contribution

In this chapter we provide a comprehensive survey on group synchronization, gathering several works that have been developed in different communities, including Computer Vision, Photogrammetry, Robotics, and Graph Theory, while at the same time also proposing some novel techniques. More precisely, our contributions are the following.

First, in Section 2.2 we set forth a theoretical unified framework where several synchronization problems are seen as instances of a more abstract principle, which is grounded on the notion of group-labelled graph.

Secondly, we provide a detailed description of direct solutions to synchronization, which are based on a matrix formulation of the problem: Section 2.3 is devoted to synchronization over $(\mathbb{R}, +)$, which is expressed as a linear system of equations; Section 2.4 addresses the synchronization problem over $(\mathbb{R} \setminus \{0\}, \cdot)$, which can be cast to a spectral decomposition or a null-space problem, and Section 2.5 generalizes such solutions to synchronization over $(GL(d), \cdot)$. Then, several subgroups of $GL(d)$ are analysed, namely $SL(d)$ (Section 2.6), $O(d)$ (Section 2.7), $SE(d-1)$ (Section 2.8), and $Sym(d)$ (Section 2.9), in which cases the solution needs to be projected onto the group, as closure is not guaranteed. Applying the spectral and null-space solutions to the Special Euclidean Group is one of the contributions of this chapter, which generalizes the works in [167, 4, 168]. Another contribution is

showing that the spectral solution can be extended to synchronization over $ISym(d)$ (Section 2.10), which is an inverse monoid and a subsemigroup of $Sym(d)$. The importance of a matrix formulation of synchronization is three-fold: this framework is theoretically appealing since it can be applied to any group admitting a matrix representation, as opposed to other techniques which are based on ad-hoc minimizations of specific cost functions; it results in fast techniques, as the synchronization problem is cast to closed-form solutions, such as spectral decomposition or linear least squares; it easily copes with weights on individual relative measures, allowing a straightforward robust extension via Iteratively Reweighted Least Squares. Some novel algebraic properties are proved in support of this formulation, namely Propositions 2.4, 2.5, 2.6, 2.7, 2.10 and 2.11.

Finally, in Section 2.5.5 we show that the synchronization problem can also be expressed in terms of "low-rank and sparse" (LRS) matrix decomposition, that is the problem of recovering a low-rank matrix starting from an incomplete subset of its entries, which are corrupted by noise and outliers. This formulation, which is unprecedented in the literature, neatly caters for missing data, outliers and noise, and it benefits from a wealth of available decomposition algorithms that can be plugged-in, such as R-GoDec, Grasta [90] or L1-Alm [202], which are reviewed in Appendix C. This framework can be successfully applied to synchronization over $SO(d)$ and $SE(d)$, as experiments in Sections 5.2 and 5.3 will show. However, it is not applicable to synchronization over $Sym(d)$ since the low-rank matrix is sparse, being composed of permutation matrices, hence it does not satisfy the incoherence assumptions (see [39, 206]) that make LRS algorithms work in practice.

2.2 Theoretical Framework

Let us start by introducing the notion of *group-labelled graph* [60]. Let $(\Sigma, *)$ be a group with unit element 1_Σ, and let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple directed graph with vertex set $\mathcal{V} = \{1, 2, \dots, n\}$ and edge set $\mathcal{E}$, with $m = |\mathcal{E}|$. We do not assume full measurements, i.e. $\mathcal{G}$ may not be complete. A Σ-*labelled* graph is a directed graph with a labelling of its edge set by elements of Σ, that is a t-uple $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ where

$$z : \mathcal{E} \to \Sigma \tag{2.1}$$

is such that if $(i, j) \in \mathcal{E}$ then $(j, i) \in \mathcal{E}$ and

$$z(j, i) = z(i, j)^{-1}. \tag{2.2}$$

Thus, we may also view $\mathcal{G}$ as an undirected graph.

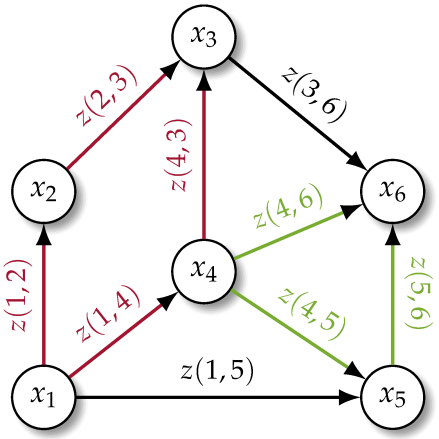
Definition 2.1. Let $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ be a Σ-labelled graph. We say that a circuit

$$\{(i_1, i_2), (i_2, i_3), \dots, (i_\ell, i_1)\} \tag{2.3}$$

is a *null cycle*[1] if and only if the composition of the edge labels along the circuit returns the identity, namely

$$z(i_1, i_2) * z(i_2, i_3) * \ldots * z(i_\ell, i_1) = 1_\Sigma. \tag{2.4}$$

The notion of null cycle is illustrated in Figure 2.2. The "null" term clearly refers to an additive notation for the group (which is used in [101, 49, 83]), without implying that Σ needs to be Abelian. Note that if the group is not commutative, then it may happen that cyclic shifts of the same circuit yield different elements of the group. Nevertheless the notion of null cycle is well defined, as either all of the cyclic shifts are equal to 1_Σ or none of them, as observed in [83]. As noted in [14], the concept of null cycle resembles Kirchoff's voltage law, stating that the electrical potential differences along any cycle sum to zero.



non-null cycle
$z(1,4) * z(4,3) * z(3,2) * z(2,1) \neq 1_\Sigma$

null cycle
$z(4,5) * z(5,6) * z(6,4) = 1_\Sigma$

FIGURE 2.2: Null and non-null cycles.

Definition 2.2. Let $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ be a Σ-labelled graph. Let $x : \mathcal{V} \to \Sigma$ be a vertex labelling. We say that x is a *consistent labelling* if and only if

$$z(e) = x(i) * x(j)^{-1} \quad \forall\, e = (i, j) \in \mathcal{E}. \tag{2.5}$$

A Σ-labelled graph admitting a consistent labelling is also called *balanced* [97]. Equation (2.5) means that each edge label is the ratio of the corresponding vertex labels, as shown in Figure 2.3. Such condition is referred to as *consistency constraint*

[1] A circuit is also a cycle; definitions are given in Appendix B.

and it is equivalent to

$$z(e) * x(j) = x(i) \quad \forall\, e = (i,j) \in \mathcal{E}. \tag{2.6}$$

It is understood that a consistent labelling is defined up to a global (right) product with any group element, in the sense that if $x : \mathcal{V} \to \Sigma$ is consistent then also $y : \mathcal{V} \to \Sigma$, $y(i) = x(i) * s$, is consistent, for any (fixed) $s \in \Sigma$.

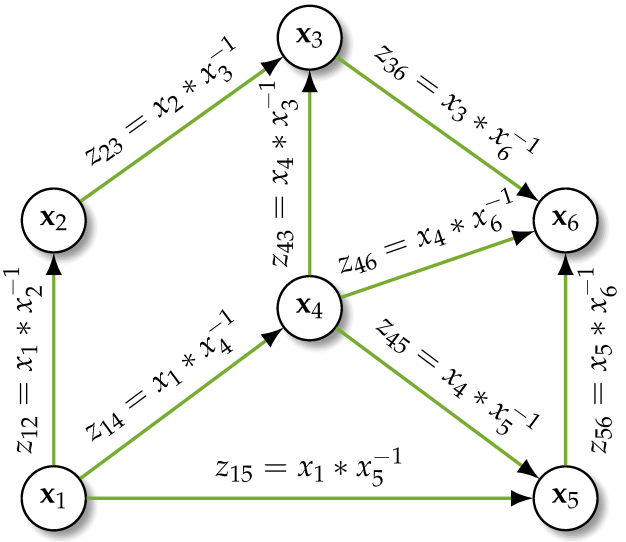


FIGURE 2.3: Consistent labelling.

The notion of consistent labelling is strictly related to that of null cycle, as stated by the following result.

Proposition 2.1 ([83]). *Let $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ be a Σ-labelled graph. There exists a polynomial algorithm which either finds a non-null cycle in Γ or finds a consistent labelling of Γ.*

The following procedure draws an outline of the proof. First, compute a spanning tree (see Appendix B) and use Equation (2.6) to label nodes, starting from the root labelled with the identity 1_Σ: this is a consistent labelling by construction. Then add one by one the edges not belonging to the spanning tree, thereby creating a circuit. If the cycle is null then the edge can be added and leave the labelling consistent, otherwise a non-null cycle has been found.

Corollary 2.1 ([83]). *The Σ-labelled graph $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ has a consistent labelling if and only if it does not contain a non-null cycle.*

2.2.1 Group Feedback Edge Set

The problem of finding non-null cycles in a group-labelled graph is studied in Graph Theory community under the name of "group feedback edge set" problem [83]. Specifically, the goal is to break non-null cycles by deleting k edges, where $k \in \mathbb{N}$ is assumed to be known.

Definition 2.3. Let $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ be a Σ-labelled graph. The *Group Feedback Edge Set* (GFES) problem is defined as follows: on input (Γ, k) for some $k \in \mathbb{N}$, decide whether there exists a subset of the edges $\mathcal{S} \subseteq \mathcal{E}$ with $|\mathcal{S}| \leq k$ such that the labelled graph of the remaining edges $\Gamma' = (\mathcal{V}, \mathcal{E} \setminus \mathcal{S}, z)$ does not contain a non-null cycle.

With some abuse of notation, in Definition 2.3 we denote with $(\mathcal{V}, \mathcal{E} \setminus \mathcal{S}, z)$ the Σ-labelled graph with edges in $\mathcal{S}$ removed from $\mathcal{E}$, even though formally z has in its domain edges that does not exist in $\mathcal{E} \setminus \mathcal{S}$.

The set $\mathcal{S}$ satisfying Definition 2.3 (if it exists) is called the *feedback edge set* of Γ. The interpretation is that $\mathcal{S}$ identifies edges with *outlying labels* that prohibit a consistent labelling to be found. Note that in the presence of noise we have to relax Equation (2.4) and consider the following

$$\delta\big(z(i_1, i_2) * z(i_2, i_3) * \ldots * z(i_\ell, i_1), 1_\Sigma\big) / \sqrt{\ell} \leq \tau \tag{2.7}$$

where $\tau \geq 0$ is a given threshold and it is assumed that Σ admits a metric function $\delta : \Sigma \times \Sigma \to \mathbb{R}^+$. The normalization factor $\sqrt{\ell}$ takes into account error propagation when considering long cycles [61].

Outlying labels can be detected through the methods in [186, 57], which come from Graph theory community. Alternatively, Computer Vision solutions can be used [61, 11, 198, 31, 80, 138]. The authors of [61] consider a maximum-weight spanning tree, and they analyze cycles formed by the remaining edges. In [11] some heuristics based on cycle bases are introduced to improve this scheme. In [198] a Bayesian framework is used to classify all the edges of the measurement graph into inliers and outliers. The authors of [134] show that an iterative use of this method can remove most outlier edges in the graph. In [31] it is assumed that there exists a spanning tree without dependent outliers (it may contain independent outliers), and an iterative approach based on a Kalman filter is developed for outlier detection. Other approaches [80, 138] are based on random spanning trees, in a RANSAC-like fashion. These strategies are computationally demanding and do not scale well with the size of the graph.

2.2.2 Group Synchronization

Let us assume that Σ is equipped with a metric function $\delta : \Sigma \times \Sigma \to \mathbb{R}^+$ and let $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ be a non-negative non-decreasing function with a unique minimum in 0 and $\rho(0) = 0$. Some instances are the quadratic loss function $\rho(y) = y^2$ or robust loss functions such as those used in M-estimators [91].

Definition 2.4. Let $\Gamma = (\mathcal{V}, \mathcal{E}, z)$ be a Σ-labelled graph. Let $\widetilde{x} : \mathcal{V} \to \Sigma$ be a vertex labelling. We define the *consistency error* of $\widetilde{x}$ as the quantity

$$\epsilon(\widetilde{x}) = \sum_{(i,j) \in \mathcal{E}} \rho\Big(\delta\big(\widetilde{z}(i,j), z(i,j)\big)\Big) \tag{2.8}$$

where $\widetilde{z}$ is the edge labelling induced by $\widetilde{x}$, namely $\widetilde{z}(i,j) = \widetilde{x}(i) * \widetilde{x}(j)^{-1}$.

A vertex labelling is consistent if and only if it has zero consistency error. In practical applications a labelling with zero error hardly exists, since the edge labels are corrupted by noise, thus the goal is to address the following problem.

Definition 2.5. Given a Σ-labelled graph $\Gamma = (\mathcal{V}, \mathcal{E}, z)$, the *group synchronization problem* consists in finding a vertex labelling with minimum consistency error.

In other words, one wants to recover the unknown group elements (vertex labels) given a redundant set of noisy measurements of their ratios (edge labels), as shown in Figure 2.4.

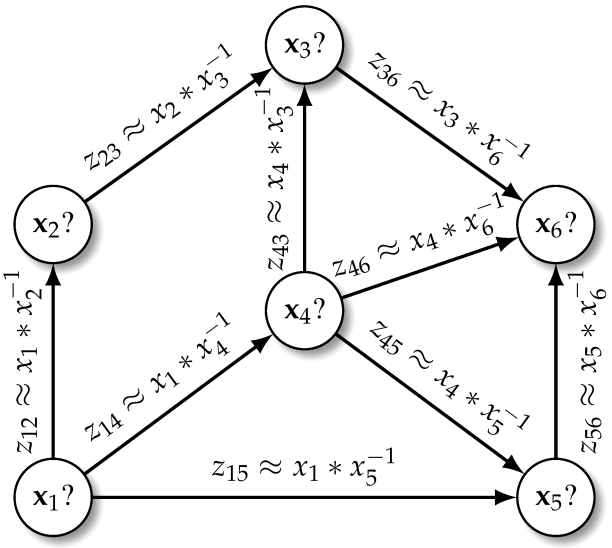


FIGURE 2.4: The synchronization problem.

According to the chosen group, we have several instances of synchronization:

- $\Sigma = \mathbb{Z}_2$ *sign synchronization* [54];
- $\Sigma = \mathbb{R}$ *time synchronization* [98, 75];
- $\Sigma = \mathbb{R}^d$ *translation synchronization* [13, 154, 179, 132, 3];
- $\Sigma = SO(d)$ *rotation synchronization* (rotation averaging) [163, 126, 167, 52, 70, 28, 29, 86, 44, 187, 7, 182];
- $\Sigma = SE(d)$ *rigid-motion synchronization* (motion averaging or pose graph optimization) [71, 79, 177, 180, 20, 10, 9, 151, 150];
- $\Sigma = SL(d)$ *homography synchronization* [159];
- $\Sigma = \mathbb{S}_d$ *permutation synchronization* [143, 164, 197, 8].

The synchronization problem requires the graph to be connected, but *error compensation* happens only with cycles. The minimum number of relative measures is $n-1$, which makes $\mathcal{G}$ a tree. In this case every vertex can be labeled by simply propagating Equation (2.6) along the tree, starting from the root labeled with the identity element. In this way, however, there is no remedy to error propagation: the error affecting an edge label propagates down to the leaves of the tree without compensation. In the synchronization problem, instead, the goal is to exploit *redundant* relative measures in a global fashion to improve the final estimate.

If the measures are also corrupted by outliers, one needs to solve a GFES problem beforehand, using a relaxed notion of null cycle, i.e., Equation (2.7). Alternatively, a robust loss function can be used in (2.8) without detecting outliers explicitly.

2.3 Synchronization over $(\mathbb{R}^d, +)$

In this section we derive a direct solution for the synchronization problem over $\mathbb{R}$ and show that such solution can be easily generalized to $\mathbb{R}^d$, following [13, 154].

2.3.1 Time Synchronization

Let us start by considering the synchronization of real numbers with the addition, namely $(\Sigma, *) = (\mathbb{R}, +)$ (also known as *time synchronization*). A vertex labelling $x : \mathcal{V} \to \mathbb{R}$ is consistent with a given edge labelling $z : \mathcal{E} \to \mathbb{R}$ if and only if[2]

$$x_i - x_j = z_{ij} \quad \forall\, (i,j) \in \mathcal{E}. \tag{2.9}$$

If we denote the *incidence vector* of the edge (i,j) with

$$\boldsymbol{b}_{ij} = [\, 0, \ldots, \underset{\uparrow}{1}_{i}, \ldots, \underset{\uparrow}{-1}_{j}, \ldots, 0\,] \tag{2.10}$$

then Equation (2.9) can be written as

$$\boldsymbol{b}_{ij}\, [\, x_1, \ldots, x_n\,]^{\mathsf{T}} = z_{ij} \quad \forall\, (i,j) \in \mathcal{E} \tag{2.11}$$

or, equivalently, in matrix form

$$B^{\mathsf{T}}\mathbf{x} = \mathbf{z} \tag{2.12}$$

where B is the $n \times m$ incidence matrix of the *directed* graph $\mathcal{G}$, which has the vectors $\boldsymbol{b}_{ij}$ as columns, $\mathbf{x} \in \mathbb{R}^n$ is the vector containing all the vertex labels, namely $\mathbf{x} = [x_1 \ldots x_n]^{\mathsf{T}}$, and $\mathbf{z} \in \mathbb{R}^m$ is the vector containing all the edge labels (ordered as in B), namely $\mathbf{z} = [z_{12} \ \ldots \ z_{ij} \ \ldots\,]^{\mathsf{T}}$. See Appendix B.2 for the definition of incidence matrix and related properties.

[2] For simplicity of notation, hereafter we will use subscripts instead of parenthesis to denote indices of a node/edge labelling.

We assume that the graph is connected, hence Equation (B.8) holds. Since the solution to the synchronization problem is defined up to a global group element, we are allowed (without loss of generality) to arbitrarily set $x_k = 0 = 1_\Sigma$ for a chosen $k \in \mathcal{V}$. Removing x_k from the unknowns and the corresponding row in B leaves a full-rank $(n-1) \times m$ matrix.

With a suitable choice of δ and ρ in Equation (2.8), the consistency error of the synchronization problem writes

$$\epsilon(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}} |x_i - x_j - z_{ij}|^2 = \|B^\mathsf{T}\mathbf{x} - \mathbf{z}\|^2 \tag{2.13}$$

where $\|\cdot\|$ denotes the Euclidean norm. Thus the least squares solution of Equation (2.12) solves the synchronization problem.

Remark 2.1. If $\mathbf{c} \in \{-1,0,1\}^m$ denotes the indicator vector of a circuit in $\mathcal{G}$, the cycle is null if and only if

$$\mathbf{c}^\mathsf{T}\mathbf{z} = 0. \tag{2.14}$$

If the equations coming from all the circuits in a (directed) cycle basis are stacked, then we get

$$C\mathbf{z} = \mathbf{0} \tag{2.15}$$

where $C \in \{-1,0,1\}^{(m-n+1)\times m}$ denotes the cycle matrix associated to the basis. See Appendix B for the definitions of cycle basis and cycle matrix.

It can be shown that the edge labels produced by the synchronization process are the closest to the input edge labels among those that yield null-cycles.

Proposition 2.2 ([154])**.** *If $\widetilde{\mathbf{x}}$ is the least-squares solution to Equation* (2.12)*, then the induced edge labelling $\widetilde{\mathbf{z}} = B^\mathsf{T}\widetilde{\mathbf{x}}$ solves the following constrained minimization problem*

$$\min_{\widetilde{\mathbf{z}}} \|\mathbf{z} - \widetilde{\mathbf{z}}\|^2 \quad \textit{s.t. } C\widetilde{\mathbf{z}} = \mathbf{0} \tag{2.16}$$

where C denotes the cycle matrix associated to a cycle basis of $\mathcal{G}$.

2.3.2 Translation Synchronization

Let us now consider the synchronization of real vectors with the addition, namely $(\Sigma, *) = (\mathbb{R}^d, +)$, which is also known as *translation synchronization*. A vertex labelling $\mathbf{x} : \mathcal{V} \to \mathbb{R}^d$ is consistent with a given edge labelling $\mathbf{z} : \mathcal{E} \to \mathbb{R}^d$ if and only if

$$\mathbf{x}_i - \mathbf{x}_j = \mathbf{z}_{ij} \quad \forall\, (i,j) \in \mathcal{E}. \tag{2.17}$$

It is easy to see (reasoning as in the scalar case) that the equations above can be collected for all the edges and expressed in matrix form as

$$XB = Z \tag{2.18}$$

where X is the $d \times n$ matrix obtained by juxtaposing all the vertex labels, namely $X = [\mathbf{x}_1 \dots \mathbf{x}_n]$, and Z is the $d \times m$ matrix obtained by juxtaposing all the edge labels, namely $Z = [\mathbf{z}_{12} \ \dots \ \mathbf{z}_{ij} \ \dots]$.

Applying the vectorization operator $\mathrm{vec}(\cdot)$ to both sides in (2.18) and using formula (A.5) we get

$$(B^\mathsf{T} \otimes I_d)\,\mathrm{vec}(X) = \mathrm{vec}(Z) \tag{2.19}$$

where I_d denotes the $d \times d$ identity matrix and $\otimes$ denotes the Kronecker product, which is defined in Appendix A. This is a generalization of Equation (2.12), where the incidence matrix B gets "inflated" by the Kronecker product with I_d in order to cope with the vector representation of the group elements.

Under the assumption that the graph is connected we have $\mathrm{rank}(B) = n - 1$ and hence, using (A.7), $\mathrm{rank}(B^\mathsf{T} \otimes I_d) = dn - d$. The rank deficiency corresponds to the translation ambiguity. By the same token as before, removing one vertex label from the unknowns and the corresponding row in B leaves a full-rank matrix.

With a suitable choice of δ and ρ, the consistency error of the synchronization problem writes

$$\epsilon(X) = \|(B^\mathsf{T} \otimes I_d)\,\mathrm{vec}(X) - \mathrm{vec}(Z)\|^2 \tag{2.20}$$

thus the least-squares solution of Equation (2.19) solves the synchronization problem, as in the case of time synchronization.

Remark 2.2. The null-cycle constraint for a circuit $\mathbf{c} \in \{-1, 0, 1\}^m$ rewrites

$$Z\mathbf{c} = \mathbf{0}. \tag{2.21}$$

If the equations coming from all the circuits in a (directed) cycle basis are stacked, then we obtain

$$ZC^\mathsf{T} = 0 \tag{2.22}$$

where $C \in \{-1, 0, 1\}^{(m-n+1)\times m}$ denotes the cycle matrix associated to the basis. Using the vectorization operator and formula (A.5), such equation can also be expressed as

$$(C \otimes I_d)\,\mathrm{vec}\,(Z) = \mathbf{0}. \tag{2.23}$$

Alternatively, the above equation can be derived by multiplying (2.19) by $C \otimes I_d$ and using property (A.4)

$$\big((CB^\mathsf{T}) \otimes I_d\big)\,\mathrm{vec}(X) = (C \otimes I_d)\,\mathrm{vec}(Z). \tag{2.24}$$

Note that $CB^\mathsf{T} = 0$ for any cycle matrix C, as stated by Equation (B.13), thus the left-side vanishes, yielding Equation (2.23).

It is straightforward to see that Proposition 2.2 extends to the case of synchronization over $\mathbb{R}^d$, since we can view each component in Equation (2.17) as a synchronization over $(\mathbb{R}, +)$.

2.3.3 Robust Synchronization

Resistance to outliers (i.e. wrong edge labels) can be obtained by replacing $\rho(y) = y^2$ in Equations (2.13) and (2.20) with another function $\rho(y)$ with sub-quadratic growth, and solving the resulting minimization problem with Iteratively Reweighted Least Squares (IRLS) [91]. This technique iteratively solves weighted least squares problems where the weights are computed at each iteration as a function of the residuals of the current solution.

Several weight functions have been proposed in the literature, that correspond to different ρ functions. Among them, the Cauchy weight function is one of the most popular

$$w_{ij} = \frac{1}{1 + \left(r_{ij}/c\right)^2} \tag{2.25}$$

where $r_{ij} = \delta\left(\widetilde{z}_{ij}, z_{ij}\right)$ and the tuning constant c is chosen so as to yield a reasonably high efficiency in the normal case, and still offer protection against outliers. In particular, $c = 2.385\sigma$ produces 95-percent efficiency when the errors are normal with standard deviation σ. The latter can be robustly estimated from the median absolute deviation (MAD) of the residuals as $\sigma = \text{MAD}/0.6745$.

Another example is the bisquare (also known as biweight) function [133] that assigns zero weight to residuals higher than a threshold

$$w_{ij} = \begin{cases} (1 - (r_{ij}/c)^2)^2 & \text{if } |r_{ij}| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

where the default value for the tuning constant is $c = 4.685\sigma$.

2.4 Synchronization over $\left(\mathbb{R} \setminus \{0\}, \cdot\right)$

We now consider the synchronization of real numbers with the multiplication, namely $(\Sigma, *) = (\mathbb{R} \setminus \{0\}, \cdot)$. A vertex labelling $x : \mathcal{V} \to \mathbb{R}$ is consistent with a given edge labelling $z : \mathcal{E} \to \mathbb{R}$ if and only if

$$z_{ij} = x_i \cdot x_j^{-1} \quad \forall\, (i,j) \in \mathcal{E}. \tag{2.27}$$

Let $\mathbf{x} \in \mathbb{R}^n$ be the vector containing the vertex labels and let $Z \in \mathbb{R}^{n \times n}$ be the matrix containing the edge labels

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & z_{12} & \dots & z_{1n} \\ z_{21} & 1 & \dots & z_{2n} \\ \dots & & & \dots \\ z_{n1} & z_{n2} & \dots & 1 \end{bmatrix}. \tag{2.28}$$

For a complete graph, the consistency constraint can be expressed in matrix form as

$$Z = \mathbf{x}\mathbf{x}^{-\mathsf{T}} \tag{2.29}$$

where $\mathbf{x}\mathbf{x}^{-\mathsf{T}}$ contains the edge labels induced by $\mathbf{x}$, and $\mathbf{x}^{-\mathsf{T}}$ denotes the row-vector containing the inverse of each vertex label, namely $\mathbf{x}^{-\mathsf{T}} = [x_1^{-1}\ x_2^{-1}\ \dots\ x_n^{-1}]$. Note that Equation (2.29) implies that $\text{rank}(Z) = 1$.

Remark 2.3. By inspection it can be verified that

$$\mathbf{x}^{-\mathsf{T}}\mathbf{x} = n \tag{2.30}$$

which implies that Z/n is idempotent.

If the graph is not complete then Z is not fully specified. In this case missing edges are represented as zero entries[3], i.e. $Z_A := Z \circ A$ represents the matrix of the available measures, where $\circ$ is the Hadamard product and A is the adjacency matrix of the graph $\mathcal{G}$. Being a matrix of 0/1, the effect of its entry-wise product with Z is to zero the unspecified entries and leave the others unchanged. See Appendices A and B for the definitions of Hadamard product and adjacency matrix, respectively. Hence the consistency constraint writes

$$Z_A = (\mathbf{x}\mathbf{x}^{-\mathsf{T}}) \circ A. \tag{2.31}$$

With a suitable choice of δ and ρ in Equation (2.8) the consistency error of the synchronization problem writes

$$\epsilon(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}} |z_{ij} - x_i \cdot x_j^{-1}|^2 = \|Z_A - (\mathbf{x}\mathbf{x}^{-\mathsf{T}}) \circ A\|_F^2 \tag{2.32}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The minimization of ϵ is a non-linear least squares problem, for which closed-form solutions do not seem to exist. However, two direct solutions to a related version of the problem exist, which can be derived by considering the exact (noiseless) case.

2.4.1 Spectral Solution

Let us consider the noiseless case, i.e. $\epsilon = 0$, and let us start assuming that the graph is complete. Using the consistency constraint and (2.30) we obtain

$$Z\mathbf{x} = n\mathbf{x} \tag{2.33}$$

which means that – in the absence of noise – $\mathbf{x}$ is the eigenvector of Z associated to the eigenvalue n. Note that, since Z has rank 1, all the other eigenvalues are zero, thus n is also the *largest* eigenvalue of Z.

[3]Please note that 0 does not belong to the group, hence it is available as a "special" value.

We now consider the case of missing edges in which the graph is not complete and the adjacency matrix comes into play.

Proposition 2.3 ([167])**.** *The vertex labelling* $\mathbf{x}$ *is the eigenvector of* $D^{-1}Z_A$ *associated to the eigenvalue* 1.

Proof. Using Equation (A.15), the consistency constraint can be expressed as

$$Z_A = \operatorname{diag}(\mathbf{x}) A \operatorname{diag}(\mathbf{x}^{-\mathsf{T}}) = \operatorname{diag}(\mathbf{x}) A \operatorname{diag}(\mathbf{x})^{-1} \tag{2.34}$$

which implies that

$$Z_A \mathbf{x} = \operatorname{diag}(\mathbf{x}) A \mathbf{1}_{n\times 1} = \operatorname{diag}(A\mathbf{1}_{n\times 1})\mathbf{x} = D\mathbf{x} \tag{2.35}$$

where $\mathbf{1}_{n\times 1}$ is a vector of ones and D is the degree matrix of the graph (see Appendix B.2). □

Note that the incomplete data matrix Z_A has full rank in general, thus 1 is not the unique non-zero eigenvalue of $D^{-1}Z_A$, in contrast to the case of a complete graph. However, it can be shown that 1 is the *largest* eigenvalue of $D^{-1}Z_A$.

Proposition 2.4. *The matrix* $D^{-1}Z_A$ *has real eigenvalues. The largest eigenvalue is 1 and it has multiplicity 1.*

Proof. Since diagonal matrices commute, it follows from Equation (2.34) that

$$D^{-1}Z_A = \operatorname{diag}(\mathbf{x})(D^{-1}A) \operatorname{diag}(\mathbf{x})^{-1} \tag{2.36}$$

hence $D^{-1}Z_A$ and $D^{-1}A$ are similar, i.e. they have the same eigenvalues. The matrix $D^{-1}A$ is the transition matrix of the graph $\mathcal{G}$ (see Appendix B.2), which – as a consequence of the Perron-Frobenius theorem – has real eigenvalues and 1 is the largest eigenvalue (with multiplicity 1), if the graph is connected. □

The proof of Proposition 2.4 has pointed out that – provided that Z is decomposable as $Z = \mathbf{x}\mathbf{x}^{-\mathsf{T}}$ – the matrix $D^{-1}Z_A$ has a particular structure that yields real eigenvalues, although it is not symmetric. In particular, the eigenvalues do not depend on the measured data, but they depend only on the structure of the graph $\mathcal{G}$ (through the matrices D and A).

When noise is present, i.e. $\epsilon \neq 0$, the eigenvector of $D^{-1}Z_A$ corresponding to the largest eigenvalue is an estimate of the vertex labelling $\mathbf{x}$. The presence of noise, however, cripples the structure of Z_A, i.e. $Z_A \neq (\mathbf{x}\mathbf{x}^{-\mathsf{T}}) \circ A$, thus the eigenvalues and the eigenvectors may be complex. As a consequence, after computing the leading eigenvector, the imaginary part is zeroed. To the best of our knowledge, no general results are known linking this spectral solution to the synchronization cost function. This approach resembles the spectral solution developed in [167] for $SO(2)$.

Note that an eigenvector is defined up to scale, and the scale indeterminacy is in agreement with the fact that the solution to synchronization is defined up to a global group element.

Remark 2.4. The top eigenvector can be computed by the power iteration method, which, considering (e.g.) the case of a complete graph, starts with a random vector $\mathbf{x}_0 \in \mathbb{R}^n$ and iterates the relation $\mathbf{x}_{k+1} = Z\mathbf{x}_k / \|Z\mathbf{x}_k\|$, thus it requires to compute Z^k, for $k = 1, 2, \ldots, k_{\max}$. It is observed in [167] that multiplying the matrix Z by itself integrates the consistency relation of triplets, while high order iterations exploit consistency relations of longer cycles. Indeed

$$\begin{aligned} Z^2_{ij} &= \sum_{k=1}^{n} Z_{ik} Z_{kj} \\ Z^3_{ij} &= \sum_{k=1}^{n} \sum_{h=1}^{n} Z_{ik} Z_{kh} Z_{hj} \ \ldots \end{aligned} \tag{2.37}$$

Thus the top eigenvector integrates the consistency relation of all cycles.

2.4.2 Null-space Solution

We now show that synchronization over $(\mathbb{R} \setminus \{0\}, \cdot)$ can also be expressed as a null-space problem. If $\epsilon = 0$ Equation (2.33) is equivalent to

$$(nI_n - Z)\mathbf{x} = \mathbf{0} \tag{2.38}$$

which means that, if the graph is complete, the vertex labelling $\mathbf{x}$ coincides with the 1-dimensional null-space of $nI_n - Z$. In the case of missing edges, let us rewrite (2.35) as

$$(D - Z_A)\mathbf{x} = \mathbf{0} \tag{2.39}$$

thus $\mathbf{x}$ belongs to the null-space of $D - Z_A$.

Let us observe that $D = D \circ Z$, since Z has ones along its diagonal and D is diagonal. Using the distributive property of the Hadamard product, we obtain an equivalent expression for $D - Z_A$

$$D - Z_A = (D - A) \circ Z = L \circ Z \tag{2.40}$$

where $L = D - A$ is the Laplacian matrix of $\mathcal{G}$ (see Appendix B.2). Note that in practice one cannot measure the matrix $L \circ Z$, since the full Z is not available. In fact, only the product $Z \circ A$ is available. Therefore, the left side in (2.40) will be used in practical scenarios. However, the right side emphasizes the presence of the Laplacian matrix, which is useful to prove that the null-space of $D - Z_A$ is 1-dimensional, as happens in the case of a complete graph.

Proposition 2.5. *The matrix $D - Z_A$ has a 1-dimensional null-space.*

Proof. Using Equation (A.15) and the consistency constraint (2.29) we get

$$D - Z_A = L \circ Z = \operatorname{diag}(\mathbf{x}) L \operatorname{diag}(\mathbf{x}^{-\mathsf{T}}) = \operatorname{diag}(\mathbf{x}) L \operatorname{diag}(\mathbf{x})^{-1} \tag{2.41}$$

which means that $D - Z_A$ and L are similar, thus they have the same rank. The rank of the Laplacian matrix is $n - 1$, under the assumption that the graph is connected (see Appendix B.2), thus we have the thesis. □

When noise is present, an estimate of $\mathbf{x}$ is given by the right singular vector of $D - Z_A$ corresponding to the least singular value. This approach solves the following problem

$$\min_{\|\mathbf{x}\|=1} \|(D - Z_A)\mathbf{x}\|^2 \tag{2.42}$$

which can also be expressed as

$$\min_{\|\mathbf{x}\|=1} \sum_{i=1}^{n} \left(\sum_j (z_{ij} x_j - x_i) \right)^2 \tag{2.43}$$

where the constraint $\|\mathbf{x}\| = 1$ fixes the global scale. For each node, this cost function considers the edges incident to that node, it sums the residuals of the compatibility constraints, and takes the square. These terms are then summed up over all the nodes. A formal relationship between this cost function and the consistency error of the synchronization problem has still to be found.

Remark 2.5. In the presence of noise the null-space and spectral solutions do not coincide, in general. In the case of a complete graph, for instance, they coincide if and only if the unique non zero eigenvalue of Z is exactly n, which is unlikely to happen in practice. Moreover, while the spectral solution computes the eigenvalue decomposition of $D^{-1}Z_A$, which is related to the null-space of $D^{-1}Z_A - I = D^{-1}(Z_A - D)$ (assuming that the largest eigenvalue is $\lambda = 1$), the null-space solution considers the matrix $Z_A - D$. This matrix has the same kernel of $D^{-1}(Z_A - D)$ only when $Z_A - D$ is rank deficient, which is not true in the presence of noise. In sloppy terms, the "approximate" kernel of $Z_A - D$ does not coincide with the "approximate" kernel of $D^{-1}(Z_A - D)$.

2.4.3 Robust Synchronization

It is easy to see that the above analysis can be extended to handle weighted measurements, which translates in letting the entries of A to assume values in $[0, 1]$, where 0 still indicates a missing measurement and the other values reflect the reliability of the edge labels. This allows a straightforward extension to gain resilience to outliers via an IRLS-like scheme, i.e. an estimate for the vertex labelling with given weights – stored in the symmetric adjacency matrix A – is obtained either from the top eigenvector of $D^{-1}Z_A$ or from the least right singular vector of $D - Z_A$, then the weights are updated through, e.g., the Cauchy weight function (2.25), and these steps are iterated until convergence or a maximum number of iterations is reached.

2.5 Synchronization over $(GL(d), \cdot)$

In this section we consider the synchronization problem over the General Linear Group $GL(d)$, which is the set of all $d \times d$ invertible matrices, where the group operation $*$ reduces to matrix multiplication and $1_\Sigma = I_d$. A vertex labelling $X : \mathcal{V} \to \mathbb{R}^{d\times d}$ is consistent with a given edge labelling $Z : \mathcal{E} \to \mathbb{R}^{d\times d}$ if and only if $Z_{ij} = X_i \cdot X_j^{-1}$.

All the vertex/edge labels can be collected in two matrices $X \in \mathbb{R}^{dn\times d}$ and $Z \in \mathbb{R}^{dn\times dn}$ respectively, which are "matrices of matrices", namely they are defined as in Equation (2.28) with the provision that each element is now a $d \times d$ matrix, and the diagonal of Z is filled with identity matrices

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_n \end{bmatrix}, \quad Z = \begin{bmatrix} I_d & Z_{12} & \dots & Z_{1n} \\ Z_{21} & I_d & \dots & Z_{2n} \\ \dots & & & \dots \\ Z_{n1} & Z_{n2} & \dots & I_d \end{bmatrix}. \tag{2.44}$$

For a complete graph, the consistency constraint rewrites

$$Z = XX^{-\flat} \tag{2.45}$$

which implies that $\operatorname{rank}(Z) = d$, where $X^{-\flat} \in \mathbb{R}^{d\times dn}$ denotes the block-matrix containing the inverse of each $d \times d$ block of X, i.e. $X^{-\flat} = [X_1^{-1}\ X_2^{-1}\ \dots\ X_n^{-1}]$.

Remark 2.6. By inspection it can be verified that

$$X^{-\flat}X = nI_d \tag{2.46}$$

and hence Z/n is idempotent.

If the graph is not complete then the available measures are given by $Z_A := Z \circ (A \otimes \mathbf{1}_{d\times d})$, where the adjacency matrix gets "inflated" by the Kronecker product with $\mathbf{1}_{d\times d}$ to match the block structure of the measures. Accordingly, the consistency constraint becomes

$$Z_A = (XX^{-\flat}) \circ (A \otimes \mathbf{1}_{d\times d}) \tag{2.47}$$

which generalizes Equation (2.31), and the synchronization cost function writes

$$\epsilon(X) = \|Z_A - (XX^{-\flat}) \circ (A \otimes \mathbf{1}_{d\times d})\|_F^2 \tag{2.48}$$

with a suitable choice of δ and ρ in Equation (2.8). The Hadamard product with $A \otimes \mathbf{1}_{d\times d}$ mirrors the summation over the edges in $\mathcal{E}$ in the definition of the consistency error. We now show that two direct solutions to a related version of the problem can be derived in the noiseless case, which are based on [167, 168, 4], generalizing the approaches detailed in Sections 2.4.1 and 2.4.2.

2.5.1 Spectral Solution

If the graph is complete, using the consistency constraint and Equation (2.46), we obtain

$$ZX = nX \tag{2.49}$$

which means that – if $\epsilon = 0$ – the columns of X are d (independent) eigenvectors of Z corresponding to the eigenvalue n. Since Z has rank d, all the other eigenvalues are zero, thus n is also the *largest* eigenvalue of Z. In the case of missing data, it can be seen that Equation (2.35) generalizes to

$$Z_A X = (D \otimes I_d) X. \tag{2.50}$$

Indeed, the i-th block-row in the above equation is

$$\sum_{j \text{ s.t. } (i,j)\in\mathcal{E}} Z_{ij} X_j = [D]_{ii} X_i \tag{2.51}$$

which is satisfied since $Z_{ij} = X_i X_j^{-1}$. Thus the columns of X are d eigenvectors of $(D \otimes I_d)^{-1} Z_A$ associated to the eigenvalue 1.

Note that the incomplete data matrix Z_A will have full rank in general, thus 1 is not the unique nonzero eigenvalue of $(D \otimes I_d)^{-1} Z_A$, in contrast to the case of Equation (2.49). However, it can be shown that 1 is the *largest* eigenvalue of such a matrix.

Proposition 2.6. *The matrix $(D \otimes I_d)^{-1} Z_A$ has real eigenvalues. The largest eigenvalue is 1 and it has multiplicity d.*

Proof. It can be seen that Equation (2.34) generalizes to

$$Z_A = \text{blkdiag}(X)(A \otimes I_d)\,\text{blkdiag}(X)^{-1} \tag{2.52}$$

where $\text{blkdiag}(X)$ produces a $dn \times dn$ block-diagonal matrix with $d \times d$ blocks $X_1, \ldots, X_n$ along the diagonal. Indeed, the (i,j)-th block in $Z_A = (XX^{-\flat}) \circ (A \otimes \mathbf{1}_{d\times d})$ is

$$\begin{cases} (X_i X_j^{-1}) \circ \mathbf{1}_{d\times d} & \text{if } (i,j) \in \mathcal{E} \\ 0 \quad \text{if } (i,j) \notin \mathcal{E} \end{cases} \tag{2.53}$$

while the (i,j)-th block in $\text{blkdiag}(X)(A \otimes I_d)\,\text{blkdiag}(X)^{-1}$ is

$$\begin{cases} X_i I_d X_j^{-1} & \text{if } (i,j) \in \mathcal{E} \\ 0 \quad \text{if } (i,j) \notin \mathcal{E}. \end{cases} \tag{2.54}$$

Note that the diagonal matrix $(D \otimes I_d)^{-1}$ commutes with $\mathrm{blkdiag}(X)$, since each $d \times d$ block along its diagonal is a multiple of the identity matrix. Thus

$$\begin{aligned}(D \otimes I_d)^{-1} Z_A = \mathrm{blkdiag}(X)(D \otimes I_d)^{-1}(A \otimes I_d)\,\mathrm{blkdiag}(X)^{-1} = \\ = \mathrm{blkdiag}(X)((D^{-1}A) \otimes I_d)\,\mathrm{blkdiag}(X)^{-1}\end{aligned} \tag{2.55}$$

where the last equality follows from properties (A.3) and (A.4). Hence $(D \otimes I_d)^{-1} Z_A$ is similar to the matrix $(D^{-1}A) \otimes I_d$, i.e., they have the same eigenvalues. The largest eigenvalue of the transition matrix $D^{-1}A$ is 1 (with multiplicity 1), if the graph is connected. Since the eigenvalues of the Kronecker product of two matrices are the product of the eigenvalues of the matrices, we conclude that the largest eigenvalue of $(D^{-1}A) \otimes I_d$ is 1 and it has multiplicity d. □

Thus the eigenvectors of $(D \otimes I_d)^{-1} Z_A$ corresponding to the d largest eigenvalues (which may be complex when $\epsilon \neq 0$) are an estimate of the vertex labelling X. Note that, since the eigenvalue 1 is repeated, the corresponding eigenvectors span a linear subspace, and hence any basis for such a space is a solution. However, a change of the basis in the eigenspace corresponds to right-multiply the eigenvectors by an invertible $d \times d$ matrix, and this agrees with the fact that the solution to synchronization is defined up to a global group element.

2.5.2 Null-space Solution

If $\epsilon = 0$ then Equation (2.49) is equivalent to

$$(nI_{dn} - Z)X = 0 \tag{2.56}$$

which means that, if the graph is complete, the vertex labelling X coincides with the d-dimensional null-space of $nI_{dn} - Z$. In the case of missing edges, let us rewrite (2.50) as

$$(D \otimes I_d - Z_A)X = 0 \tag{2.57}$$

thus X belongs to the null-space of $D \otimes I_d - Z_A$.

Let us observe that the matrix $D \otimes I_d$ coincides with $(D \otimes \mathbf{1}_{d\times d}) \circ Z$, since Z has identity blocks along its diagonal and $D \otimes \mathbf{1}_{d\times d}$ is block-diagonal. Using the distributive property of the involved products, we obtain an equivalent expression for $D \otimes I_d - Z_A$

$$D \otimes I_d - Z_A = ((D - A) \otimes \mathbf{1}_{d\times d}) \circ Z = (L \otimes \mathbf{1}_{d\times d}) \circ Z \tag{2.58}$$

where the Laplacian matrix $L = D - A$ gets inflated to a $d \times d$-block structure by the Kronecker product with $\mathbf{1}_{d\times d}$, to match the block structure of Z.

Note that in practice the left side in (2.58) will be used, since only the product $Z \circ (A \otimes \mathbf{1}_{d\times d})$ is available, and one cannot measure the matrix $(L \otimes \mathbf{1}_{d\times d}) \circ Z$.

However, the right side in (2.58) emphasizes the presence of the Laplacian matrix, which is useful to prove the following result.

Proposition 2.7. *The matrix $D \otimes I_d - Z_A$ has a d-dimensional null-space.*

Proof. It can be seen that Equation (2.41) generalizes to

$$D \otimes I_d - Z_A = (L \otimes \mathbf{1}_{d\times d}) \circ Z = \text{blkdiag}(X)(L \otimes I_d)\,\text{blkdiag}(X)^{-1} \quad (2.59)$$

which means that $D \otimes I_d - Z_A$ and $L \otimes I_d$ are similar, thus they have the same rank. The rank of the Laplacian matrix is $n-1$, under the assumption that the graph is connected. Since the rank of the Kronecker product of two matrices is the product of the rank of the matrices, we obtain

$$\text{rank}(D \otimes I_d - Z_A) = \text{rank}(L)\,\text{rank}(I_d) = dn - d \quad (2.60)$$

thus we have the thesis. □

When noise is present, an estimate of X is given by the right singular vectors of $D \otimes I_d - Z_A$ corresponding to the d least singular values, which solve the following problem

$$\min_{X^\mathsf{T}X = nI_d} \|(D \otimes I_d - Z_A)X\|_F^2. \quad (2.61)$$

2.5.3 Spectral Solution versus Null-space Solution

As observed in remark 2.5, in the presence of noise the null-space and spectral solutions do not coincide, in general. An empirical comparison between the two approaches is reported in Section 5.3 for the $\Sigma = SE(3)$ case, where it is shown that the spectral solution achieves the same accuracy as the null-space method but it is faster. In particular, note that the final rounding step, i.e. zeroing the imaginary part of the eigenvectors, do not compromise the accuracy achieved by the spectral method. However, the meaning of this step in terms of synchronization is not known and will be explored by future research.

Note that in both cases the matrices inherit the same sparsity pattern as the adjacency matrix A, thus sparse solvers can be exploited, e.g., `eigs` for the spectral method and `svds` for the null-space solution, with reference to Matlab programming. As a matter of fact, `svds(F)` calls `eigs([0 F; F' 0])`, as reported in the function documentation, and consequently it runs more slowly, for the dimension of the matrix doubles. From the computational complexity point of view, the Lanczos method (implemented by `eigs`) is nearly linear, meaning that, if the matrix is sparse, every iteration is linear in n [77], but the number of iterations cannot be bounded by a constant.

2.5.4 Additive Solution

We observe that synchronization over the General Linear Group can be cast to a translation synchronization, exploiting the fact that $GL(d)$ has the structure of a Lie group [185], where the associated Lie algebra consists of all $d \times d$ real matrices with the commutator operator serving as the Lie bracket, namely $[Y, W] = YW - WY$. Informally, a Lie group can be locally viewed as topologically equivalent to a vector space, and the local neighbourhood of any group element can be adequately described by its tangent space, whose elements form a Lie algebra. The Lie algebra and the Lie group are related by the exponential mapping, and the inverse mapping from the Lie group to the Lie algebra is given by matrix logarithm.

By taking the logarithm, the consistency constraint of the synchronization problem over $GL(d)$, that is $Z_{ij} = X_i X_j^{-1}$, can be transformed into the consistency constraint of an additive group, namely

$$\log(Z_{ij}) = \log(X_i) - \log(X_j) \tag{2.62}$$

assuming that the each of the above matrices admits a unique real logarithm. Specifically, by vectorizing each side in (2.62), a relation of the form (2.17) is obtained, which defines a translation synchronization problem. Thus the solution can be found by solving a linear system in the least-squares sense, as done in Section 2.3.2, or via IRLS (to gain robustness to outliers), as explained in Section 2.3.3. In other words, the synchronization problem is addressed in the Lie algebra rather than in the group. This approach was introduced in [79, 80, 44].

However, as observed in [79], the Euclidean distance in the Lie algebra does not coincide with the Riemannian distance in the group, but it constitutes a first-order approximation, as stated by the Baker-Campbell-Hausdorff formula [185]. For this reason, in [79, 80, 44] the solution is found by iterating between solving the linear system in the Lie algebra and remapping onto the group.

2.5.5 Low-rank Solution

We now show that synchronization over $GL(d)$ can also be expressed as a low-rank and sparse (LRS) matrix decomposition (see Appendix C), paving the way to the application of matrix decomposition techniques to synchronization problems in Computer Vision. Let $\Omega = A \otimes \mathbf{1}_{d\times d}$ denote the *pattern* (or *sampling set*) of Z_A, that is the index set of available entries, so that $Z_A = Z \circ \Omega = \mathcal{P}_\Omega(Z)$. Using this notation, the synchronization problem can be expressed as

$$\begin{gathered} \min_{X\in GL(d)^n} \epsilon(X) = \min_{X\in GL(d)^n} \|(Z - XX^{-\flat}) \circ \Omega\|_F^2 \iff \\ \min_{\widetilde{Z}} \|(Z - \widetilde{Z}) \circ \Omega\|_F^2 \quad s.t.\ \widetilde{Z} = XX^{-\flat} \end{gathered} \tag{2.63}$$

where the problem of finding a consistent vertex labelling X is reduced to that of finding an edge labelling $\widetilde{Z}$ induced by X.

If the *rank relaxation* is adopted [11], i.e. the optimization variable is enforced to have rank (at most) d, then Problem (2.63) becomes

$$\min_{L} \|(Z - L) \circ \Omega\|_F^2 \quad s.t.\ \operatorname{rank}(L) \leq d \tag{2.64}$$

where the notation L instead of $\widetilde{Z}$ underlines that L will not coincide with $XX^{-\flat}$ in general, due to the relaxation. Indeed, a matrix of rank up to r admits a decomposition of the form $L = XY^{\mathsf{T}}$, where X and Y are of r columns and r rows respectively, but the constraint $Y^{\mathsf{T}} = X^{-\flat}$ is not enforced. Problem (2.64) is a *matrix completion* problem (see Appendix C.2), which can be solved via (e.g.) the OPTSPACE algorithm [105]. Any block-column of the resulting matrix can be taken as an estimate of the unknown vertex labelling, as we already know that the solution is up to a global group element.

In order to handle outliers, a *robust matrix completion* framework can be considered instead of (2.64), namely

$$\begin{aligned} &\min_{L} \|(Z - L) \circ \Omega - S\|_F^2 \\ &s.t.\ \operatorname{rank}(L) \leq d,\ S \text{ is sparse in } \Omega \end{aligned} \tag{2.65}$$

where the additional variable S represents outliers, which are sparse over the measurement graph (by assumption). Problem (2.65) is a LRS matrix decomposition with unspecified entries and outliers, since it is associated to the formulation (C.14), namely $Z \circ \Omega = L \circ \Omega + S + N$. Thus the unknown low-rank matrix can be recovered by means of any algorithm that computes such decomposition, such as GRASTA [90] or L1-ALM [202], as explained in Appendix C.3. Alternatively, the equivalent formulation (C.19) can be used, namely $Z \circ \Omega = L + S_1 + S_2 + N$ which can be computed via R-GODEC (Algorithm 3), which, in addition, takes into account the block structure of the data matrix.

2.5.6 Synchronization over Matrix Groups

The analysis carried out in this section can be extended to the case where Σ is a subgroup of $GL(d)$, i.e., it can be embedded in $\mathbb{R}^{d\times d}$, where the group operation $*$ reduces to matrix multiplication and $1_\Sigma = I_d$. In this case the low-rank property and Propositions 2.6, 2.7 still hold, thus the unknown vertex labelling X can be recovered either from the top d eigenvectors of $(D \otimes I_d)^{-1} Z_A$ or from the least d right singular vectors of $D \otimes I_d - Z_A$. Alternatively, the low-rank solution can be exploited. Note that closure is not always guaranteed: depending on the structure of the group, the solution might need to be projected onto Σ. Some instances are $SL(d)$, $O(d)$, $SE(d-1)$, and $Sym(d)$, which will be analysed in details in the next sections. They basically differ from each other by the projection stage.

With reference to the additive solution, particularly interesting are the $\Sigma = SO(3)$ and $\Sigma = SE(3)$ cases, where the associated Lie algebras are described by 3 and 6 parameters, respectively, and the exponential and logarithm maps admit closed form expressions [131, 40]. Note that this method provides an *intrinsic* estimate, thus projection onto Σ is not required.

2.6 Synchronization over $SL(d)$

Let us consider the Special Linear Group $SL(d)$, that is the set of $d \times d$ matrices with unit determinant

$$SL(d) = \{S \in \mathbb{R}^{d \times d} \; s.t. \; \det(S) = 1\}. \tag{2.66}$$

Synchronization over $SL(3)$ is studied in [159] within the context of multiple-view homography estimation, that is why the problem is referred to as *homography synchronization*.

Since $SL(d)$ is a subgroup of $GL(d)$, a direct solution to the synchronization problem can be found either via the spectral solution, which computes the top d eigenvectors of $(D \otimes I_d)^{-1} Z_A$, or via the null-space solution, which computes the least d right singular vectors of $D \otimes I_d - Z_A$. Alternatively, the (approximate) null-space of $I_{dn} - (D \otimes I_d)^{-1} Z_A$ can be computed, as done in [159].

Let U be the $dn \times d$ matrix containing either the output of the spectral method or the null-space solution. In order to obtain elements of $SL(d)$ from U, each $d \times d$ block in U, denoted by U_i, must be scaled to unit determinant, which can be done by dividing U_i by $\sqrt[d]{\det(U_i)}$. However, if $\det(U_i)$ is negative and d is even, real roots do not exist; in this case the determinant can be always made positive by exchanging two columns of U.

2.7 Synchronization over $O(d)$

Let us consider the Orthogonal Group $O(d)$, that is the set of orthogonal transformations in d-space, which admits a matrix representation through $d \times d$ orthogonal matrices

$$O(d) = \{R \in \mathbb{R}^{d \times d} \; s.t. \; R^{\mathsf{T}} R = R R^{\mathsf{T}} = I_d\}. \tag{2.67}$$

An important subgroup of $O(d)$ is the Special Orthogonal Group $SO(d)$, that is the set of orthogonal matrices with determinant 1, which represent rotations in d-space. Synchronization over $SO(d)$ is also known as *rotation (angular) synchronization* [167] or *multiple rotation averaging* [86, 194]. A comprehensive survey on existing solutions can be found in [182].

From the theoretical perspective, synchronization over $SO(3)$ is analyzed in depth in [86, 194]. In [86] the consistency error (2.8) is studied under the choice $\rho(y) = y^p$ (with $p \geq 1$) and several distance measures are considered, including

quaternion, angular (geodesic) and chordal distances, where each metric is related to a particular parametrization of the rotation space. In [194] it is shown that smaller and well-connected graphs are easier than larger and noisy ones, based on a local convexity analysis. Further theoretical analysis is reported in [29] where Cramèr-Rao bounds for synchronization over $SO(d)$ are derived, namely lower bounds on the variance of unbiased estimators, assuming a certain noise model.

Note that the $\Sigma = O(d)$ case is a special one, since the inverse reduces to matrix transposition, thus the consistency constraint rewrites $Z_{ij} = X_i X_j^{\mathsf{T}}$ which, if the graph is complete, is equivalent to

$$Z = XX^{\mathsf{T}}. \tag{2.68}$$

Such a decomposition implies that, if $\epsilon = 0$, the matrix Z is *symmetric and positive semidefinite*, besides being low-rank. In the case of missing edges, the consistency constraint translates into

$$Z_A = (XX^{\mathsf{T}}) \circ (A \otimes \mathbf{1}_{d\times d}) \tag{2.69}$$

and the consistency error of the synchronization problem becomes

$$\epsilon(X) = \sum_{(i,j)\in\mathcal{E}} \|Z_{ij} - X_i X_j^{\mathsf{T}}\|_F^2 = \|Z_A - (XX^{\mathsf{T}}) \circ (A \otimes \mathbf{1}_{d\times d})\|_F^2. \tag{2.70}$$

As observed in Section 2.5.6, synchronization over the Orthogonal Group can be addressed either via the spectral method or the null-space solution, since $O(d)$ is a subgroup of $GL(d)$. A related approach is adopted in [126], where the $\Sigma = SO(3)$ case is considered and the least eigenvectors of $D \otimes I_3 - Z_A$ are computed (instead of the least right singular vectors). More details about this technique can be found in [126, 182].

With reference to the spectral method, it turns out that $(D \otimes I_d)^{-1} Z_A$ admits an orthonormal basis of real eigenvectors even in the presence of noise, as a consequence of the following remark.

Remark 2.7. Note that both Z and Z_A are symmetric even in the presence of noise, since, by assumption, a group-labelled graph satisfies $Z_{ji} = Z_{ij}^{-1}$ for all $(i,j) \in \mathcal{E}$, which becomes $Z_{ji} = Z_{ij}^{\mathsf{T}}$ in the $\Sigma = O(d)$ case. The matrix $(D \otimes I_d)^{-1} Z_A$ is not symmetric, but it is similar to the *symmetric* matrix $(D \otimes I_d)^{-1/2} Z_A (D \otimes I_d)^{-1/2}$, thus its eigenvalues and eigenvectors are real.

Thus the final rounding step which is required for $GL(d)$, i.e. zeroing the imaginary part of the eigenvectors, is not necessary here. Accordingly, a change of the (orthonormal) basis in the eigenspace corresponds to right-multiply the eigenvectors by an *orthogonal* $d \times d$ matrix, and this agrees with the fact that the solution to synchronization is defined up to an element of $O(d)$. The spectral method was introduced in [167] for $\Sigma = SO(2)$ and extended in [168, 4] to $\Sigma = SO(3)$.

Optimization problem

We now explain the link between the spectral solution and the consistency error of the synchronization problem, following the reasoning reported in [4].

Let us start with the case of a complete graph. Since the Frobenius norm of a matrix can be defined in terms of its trace, Equation (2.70) can be expressed as

$$\begin{aligned}\epsilon(X) &= \sum_{i,j=1}^{n} \operatorname{trace}(Z_{ij}^{\mathsf{T}} Z_{ij}) + \operatorname{trace}(X_j X_i^{\mathsf{T}} X_i X_j^{\mathsf{T}}) - 2\operatorname{trace}(X_i^{\mathsf{T}} Z_{ij} X_j) = \\ &= \sum_{i,j=1}^{n} \operatorname{trace}(Z_{ij}^{\mathsf{T}} Z_{ij}) + d - 2\operatorname{trace}(X_i^{\mathsf{T}} Z_{ij} X_j)\end{aligned} \tag{2.71}$$

where the last equality holds since $X_i^{\mathsf{T}} X_i = I_d = X_j^{\mathsf{T}} X_j$. Therefore

$$\min_{X \in O(d)^n} \epsilon(X) \Longleftrightarrow \max_{X_1,\ldots,X_n \in O(d)} \sum_{i,j=1}^{n} \operatorname{trace}(X_i^{\mathsf{T}} Z_{ij} X_j) = \max_{X \in O(d)^n} \operatorname{trace}(X^{\mathsf{T}} Z X). \tag{2.72}$$

Let us consider the *spectral relaxation*, namely the constraints over the optimization variable are relaxed and the following generalized Rayleigh problem is considered

$$\max_{X^{\mathsf{T}} X = n I_d} \operatorname{trace}(X^{\mathsf{T}} Z X) \tag{2.73}$$

where the columns of X are enforced to be orthogonal rather than imposing that each $d \times d$ block in X belongs to $O(d)$.

Proposition 2.8 ([4, 159])**.** *Equation* (2.73) *admits a closed-form solution, which is given by the* d *leading eigenvectors of* Z*.*

Proof. Let $\mathcal{F}$ be the unconstrained cost function corresponding to problem (2.73), namely

$$\mathcal{F}(X) = \operatorname{trace}(X^{\mathsf{T}} Z X) + \operatorname{trace}(\Lambda(X^{\mathsf{T}} X - n I_d)) \tag{2.74}$$

where $\Lambda \in \mathbb{R}^{d \times d}$ is a symmetric matrix of unknown Lagrange multipliers. Setting to zero the partial derivatives of $\mathcal{F}$ with respect to X we obtain

$$\frac{\partial \mathcal{F}}{\partial X} = 2ZX + 2X\Lambda = 0 \Rightarrow ZX = -X\Lambda. \tag{2.75}$$

Let $\mathbf{u_i}$ denote d eigenvectors of Z (normalized so that $\|\mathbf{u_i}\| = \sqrt{n}$) for $i = 1, \ldots, d$ and let λ_i be the corresponding eigenvalues. Then $X = [\mathbf{u_1}\ \mathbf{u_2}\ \ldots\ \mathbf{u_d}]$ satisfies both (2.75) and the constraint $X^{\mathsf{T}} X = n I_d$, with $\Lambda = -\operatorname{diag}([\lambda_1, \lambda_2, \ldots, \lambda_d])$. In other words, any set of d eigenvectors is a stationary point for the objective function $\mathcal{F}$. The maximum is attained in (2.73) if $\mathbf{u_i}$ are the eigenvectors of Z corresponding to the d largest eigenvalues. □

If the graph is not complete, we consider a different definition for the consistency error

$$\epsilon(X) = \sum_{(i,j)\in\mathcal{E}} [D]_{ii}^{-1} \|Z_{ij} - X_i X_j^{\mathsf{T}}\|_F^2 \tag{2.76}$$

in which the term associated to the edge (i,j) is weighted with the inverse of the degree of node i. Reasoning as before we get

$$\begin{aligned} \min_{X\in O(d)^n} \epsilon(X) \iff \max_{X_1,\dots,X_n\in O(d)} \sum_{(i,j)\in\mathcal{E}} [D]_{ii}^{-1} \mathrm{trace}(X_i^{\mathsf{T}} Z_{ij} X_j) = \\ = \max_{X\in O(d)^n} \mathrm{trace}(X^{\mathsf{T}} (D\otimes I_d)^{-1} Z_A X) \end{aligned} \tag{2.77}$$

which, if the spectral relaxation is adopted, becomes

$$\max_{X^{\mathsf{T}}X = nI_d} \mathrm{trace}(X^{\mathsf{T}} (D\otimes I_d)^{-1} Z_A X) \tag{2.78}$$

whose solution is given by the d leading eigenvectors of $(D\otimes I_d)^{-1} Z_A$. Note that this result is due to the special structure of $O(d)$ and is not available for synchronization over $GL(d)$.

Projection onto $O(d)$

Both the spectral method, which solves Problem (2.78), and the null-space approach, which solves Problem (2.61), are algebraic solutions, which do not enforce the orthogonality constraints that matrices in $O(d)$ should satisfy. In other words, they produce an *extrinsic* estimate of the vertex labelling X, which is suboptimal with respect to working *in* the group.

As a consequence, after computing the d leading eigenvectors of $(D\otimes I_d)^{-1} Z_A$ (or the least d right singular vectors of $D\otimes I_d - Z_A$), which are collected in a $dn\times d$ matrix U, each $d\times d$ block of U has to be projected onto $O(d)$. Such projection can be performed via Singular Value Decomposition (SVD) [102]. Specifically, if $Q\in\mathbb{R}^{d\times d}$ is a given matrix, then the nearest orthogonal matrix (in the Frobenius norm sense) is given by

$$R = UV^{\mathsf{T}} \in O(d) \tag{2.79}$$

where $Q = USV^{\mathsf{T}}$ denotes the singular value decomposition of Q. If the synchronization problem in $\Sigma = SO(d)$ is considered, the projection step is slightly different. Specifically, the nearest rotation matrix (in the Frobenius norm sense) is given by

$$R = U\,\mathrm{diag}([1,\dots,1,\det(UV^{\mathsf{T}})])V^{\mathsf{T}} \in SO(d). \tag{2.80}$$

Other relaxations

The spectral and null-space solutions follow the custom pattern to relax the constraints and subsequently project onto $O(d)$. Indeed, solving the synchronization

problem over $O(d)$ is difficult since the feasible set is non-convex. Moreover, it is shown in [86] that, in the $\Sigma = SO(3)$ case, the cost function may have multiple local minima in different regions of attraction. Other examples of relaxations include rank relaxation, which is explained in Section 2.5.5, and semidefinite programming [167, 168, 4, 187], which is briefly reviewed here.

If $\Omega = A \otimes \mathbf{1}_{d\times d}$ denotes the pattern of Z_A, then the synchronization problem can be expressed as

$$\begin{aligned} \min_{X\in O(d)^n} \epsilon(X) = \min_{X\in O(d)^n} \|(Z - XX^\mathsf{T}) \circ \Omega\|_F^2 \iff \\ \min_{\widetilde{Z}} \|(Z - \widetilde{Z}) \circ \Omega\|_F^2 \quad s.t.\ \widetilde{Z} = XX^\mathsf{T} \end{aligned} \tag{2.81}$$

where the problem of finding a consistent vertex labelling X is reduced to that of finding an edge labelling $\widetilde{Z}$ induced by X, as done in Section 2.5.5.

If the *semidefinite relaxation* is employed [167, 168, 4], i.e. the optimization variable $\widetilde{Z}$ is constrained to be symmetric positive semidefinite ($\widetilde{Z} \succeq 0$) with identity blocks along its diagonal (while the remaining properties on $\widetilde{Z}$ are not enforced), then Problem (2.81) reduces to a semidefinite program

$$\min_{\widetilde{Z}} \|(Z - \widetilde{Z}) \circ \Omega\|_F^2 \quad s.t.\ \widetilde{Z} \succeq 0,\ \widetilde{Z}_{ii} = I_d \tag{2.82}$$

which can be solved (e.g.) though interior point methods. In [187] the ℓ_1-norm is used in (2.82) in place of the ℓ_2-norm, exploiting the fact that the former is more robust to outliers than the latter, and the resulting cost function is minimized through the alternating direction augmented Lagrangian method. The authors of [187] focus on accuracy rather than efficiency, providing theoretical results about exact and stable recovery of rotations.

2.8 Synchronization over $SE(d)$

Let us consider the Special Euclidean Group $SE(d)$, that is the set of direct isometries (or rigid motions) in d-space, which admits a matrix representation through $(d+1)\times(d+1)$ matrices

$$SE(d) = \left\{ \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix},\ s.t.\ R \in SO(d), \mathbf{t} \in \mathbb{R}^d \right\}. \tag{2.83}$$

Synchronization over $SE(d)$ is also known as *rigid-motion synchronization* [10] or *motion averaging* [79, 81] or *pose-graph optimization* [41].

Since $SE(d)$ is a subgroup of $GL(d+1)$, the synchronization problem can be addressed either via the spectral method or the null-space solution. The former computes the top $d+1$ eigenvectors of $(D \otimes I_{d+1})^{-1} Z_A$ (which may be complex in the presence of noise), whereas the latter computes the least $d+1$ right singular vectors

of $D \otimes I_{d+1} - Z_A$. This approach is one of the contributions of this thesis and it can be regarded as the extension of the rotation synchronization approach introduced in [167, 168, 4].

Projection onto $SE(d)$

Both the spectral method and the null-space approach are algebraic solutions, which do not enforce constraints that matrices in $SE(d)$ should satisfy.

Let us consider (e.g.) the null-space solution. Note that any basis U for the null-space of $D \otimes I_{d+1} - Z_A$ will not coincide with the vertex labelling X even in the absence of noise, since it will not be composed of rigid motions. Specifically, it will not coincide with $[\mathbf{0}_{1\times d}\ 1]$ in rows multiple of $d+1$. In order to recover X from U it is sufficient to choose a different basis for the null-space of $D \otimes I_{d+1} - Z_A$ that satisfies such constraint, which can be found by taking a suitable linear combination of the columns of U.

More precisely, let $F \in \mathbb{R}^{n\times(d+1)n}$ be the 0/1-matrix such that $FU \in \mathbb{R}^{n\times(d+1)}$ consists of the rows of U with indices multiple of $d+1$. The coefficients $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{d+1}$ of the linear combination are solution of

$$FU\mathbf{a} = \mathbf{0}_{n\times 1}, \quad FU\mathbf{b} = \mathbf{1}_{n\times 1} \tag{2.84}$$

where the first equation has a d-dimensional solution space. Let $\mathbf{a}_1, \ldots, \mathbf{a}_d$ be a basis for the null-space of FU. Thus the columns of X corresponding to rotations coincide (up to a permutation) with $[U\mathbf{a}_1, \ldots, U\mathbf{a}_d]$ and X is recovered as $X = U[\mathbf{a}_1, \ldots, \mathbf{a}_d, \mathbf{b}]$.

In the presence of noise, Equation (2.84) is solved in the least-squares sense. Then, such a solution is projected onto $SE(d)$ – as in [17] – by forcing the rows multiple of $d+1$ to $[\mathbf{0}_{1\times d}\ 1]$ and projecting $d \times d$ rotation blocks onto $SO(d)$ through Singular Value Decomposition. This procedure also applies to the spectral solution, in which case (after computing the linear combination explained above) the imaginary part of the eigenvectors is zeroed.

Note that an *extrinsic* estimate is provided in this way, for the rigid motion constraints are relaxed to compute the solution. The idea of relaxing rigid-motion constraints is also present in [151], where the feasible set $SE(d)$ is relaxed to its convex hull, which admits a semidefinite representation [156]. This results in a convex cost function which is (globally) minimized through the interior-point method. Such a solution is then improved by (locally) minimizing the objective function over $SE(3)$ through the Levenberg-Marquardt algorithm. Another possibility is the rank relaxation described in Section 2.5.5.

Two-step synchronization

Since the Special Euclidean Group is the the semi-direct product of $SO(d)$ and $\mathbb{R}^d$, synchronization over $SE(d)$ can be alternatively addressed by breaking the problem

into rotation and translation, and solving the two sub-problems separately.

Let $X_i \in SE(d)$ denote the (unknown) vertex label of node i

$$X_i = \begin{bmatrix} R_i & \mathbf{t}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{2.85}$$

where $R_i \in SO(d)$ and $\mathbf{t}_i \in \mathbb{R}^d$ denote the rotation and translation components of the rigid motion, respectively. Similarly, each edge label $Z_{ij} \in SE(d)$ can be expressed as

$$Z_{ij} = \begin{bmatrix} R_{ij} & \mathbf{t}_{ij} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{2.86}$$

with $R_{ij} \in SO(d)$ and $\mathbf{t}_{ij} \in \mathbb{R}^d$. Using this notation, the consistency constraint for synchronization over $SE(d)$, namely $Z_{ij} = X_i X_j^{-1}$, can be equivalently rewritten as

$$R_{ij} = R_i R_j^\mathsf{T} \tag{2.87}$$

$$\mathbf{t}_{ij} = -R_i R_j^\mathsf{T} \mathbf{t}_j + \mathbf{t}_i. \tag{2.88}$$

Note that Equation (2.87) defines a rotation synchronization problem, thus the rotation components of the unknown vertex labels can be recovered as explained in Section 2.7. Equation (2.88) can be equivalently written as

$$-R_i^\mathsf{T} \mathbf{t}_{ij} = R_j^\mathsf{T} \mathbf{t}_j - R_i^\mathsf{T} \mathbf{t}_i = \mathbf{x}_i - \mathbf{x}_j \tag{2.89}$$

using the substitution $\mathbf{x}_i = -R_i^\mathsf{T} \mathbf{t}_i$. Thus – assuming that rotations have been computed beforehand – recovering the translation components of the vertex labels can be reduced to a translation synchronization problem (as defined in Section 2.3.2), where the edge labels are given by $\mathbf{z}_{ij} = -R_i^\mathsf{T} \mathbf{t}_{ij}$.

2.9 Synchronization over $Sym(d)$

Let us consider the Symmetric Group $Sym(d)$, that is the set of bijections between d objects, which admits a matrix representation through $d \times d$ permutation matrices, that is why synchronization over $Sym(d)$ is also known as *permutation synchronization* [143].

Definition 2.6. A matrix P is said to be a *permutation matrix* if exactly one entry in each row and column is equal to 1 and all other entries are 0.

Since $Sym(d)$ is a subgroup of $O(d)$, such a problem can be addressed as follows: first, all the edge labels are collected in a matrix Z_A; then, the top eigenvectors of $(D \otimes I_d)^{-1} Z_A$ are computed (spectral solution) or the least right singular vectors of $D \otimes I_d - Z_A$ are computed (null-space solution). Note that the spectral solution returns real eigenvectors and solves Problem (2.78). This approach was introduced

in [143] for a complete graph (based on [167]) and subsequently extended in [164] to the case of missing data.

Remark 2.8. Note that the low-rank solution introduced in Section 2.5.5 is not suitable for synchronization over $Sym(d)$. Indeed, the matrix Z containing all the edge labels, besides being low-rank, is *sparse* in the $\Sigma = Sym(d)$ case, being composed of permutation matrices, and hence it does not satisfy the incoherence assumptions (see [39]) which make matrix completion algorithms work in practice.

Projection onto $Sym(d)$

Let us consider the spectral method and suppose that the eigenvectors of $(D \otimes I_d)^{-1} Z_A$ corresponding to the d largest eigenvalues are collected in a $nd \times d$ matrix U. Note that U is not uniquely determined. Indeed, due to Proposition 2.6, the non-zero eigenvalues of $(D \otimes I_d)^{-1} Z_A$ are repeated, thus the corresponding eigenvectors span a linear subspace, and hence any (orthogonal) basis for such a space is a solution. However, the solution to permutation synchronization is *not* invariant to (right) multiplication by an element of $O(d)$. Indeed, it is defined up to an element of $Sym(d)$.

Therefore U is not necessarily equal to the vertex labelling X. Specifically, it will not be composed of permutation matrices, in general. So we have to face the problem of how to select, among the infinitely many Us, the one that resembles X, a matrix composed of permutation matrices. A key observation is reported in the following result, suggesting that such a problem can be solved via clustering techniques.

Proposition 2.9. *Let U denote the $nd \times d$ matrix composed by the d leading eigenvectors of $(D \otimes I_d)^{-1} Z_A$; then U has d different rows (in the absence of noise).*

Proof. Since U and X are (orthogonal) eigenvectors corresponding to the same eigenvalue, there exists an orthogonal matrix $Q \in \mathbb{R}^{d \times d}$ representing a change of basis in the eigenspace of $\lambda = 1$, such that $U = XQ$. Note that the rows of X are the rows of I_d. Since Q is invertible (hence injective), $U = XQ$ has d different rows as well. □

In the presence of noise, an estimate of the vertex labelling X can be obtained by clustering the rows of U into d clusters (e.g. with k-means), and arbitrarily assigning each centroid to a row of I_d. This arbitrary assignment corresponds to the fact that the solution to permutation synchronization is defined up to a global group element.

This procedure, however, may not return valid permutation matrices. Indeed, since there are no constraints in the clustering phase, it may happen that different rows of a $d \times d$ block in U are assigned to the same cluster, resulting in more than one entry per column equal to 1. For this reason, for each $d \times d$ block in U a permutation matrix that best maps such block into the set of centroids has to be computed (e.g. via the Kuhn-Munkres algorithm [110]), and such permutation is output as the sought solution. This procedure also applies to the null-space solution.

Remark 2.9. As observed (e.g.) in [47], permutation synchronization can be seen as a graph clustering problem, where each cluster corresponds to one object out of d. The underlying graph is constructed as follows: each vertex represents an object in a node, and edges encode the correspondences. Note that the matrix Z defined in (2.90) coincides with the adjacency matrix of such a graph. Our procedure first constructs a $dn \times d$ matrix U from the d leading eigenvectors of Z, and then applies k-means to the rows of U. This coincides with solving the permutation synchronization problem via *spectral clustering* applied to the adjacency matrix, rather than to the Laplacian matrix as customary.

2.10 Synchronization over $ISym(d)$

Let us consider the Symmetric Inverse Semigroup $ISym(d)$, that is the set of bijections between (different) subsets of d objects, which admits a matrix representation through $d \times d$ partial permutation matrices, that is why synchronization over $ISym(d)$ is referred here to as *partial permutation synchronization*.

Definition 2.7. A matrix P is said to be a *partial permutation matrix* if it has at most one nonzero entry in each row and column, and these nonzero entries are all 1.

The set $ISym(d)$ is an inverse monoid with respect to matrix multiplication (see Definition 2.8) and a subsemigroup of $Sym(d)$, where the inverse is given by matrix transposition. Despite the group structure is missing, we show that a spectral solution can be derived. Note that $0 \in ISym(d)$, thus we can not distinguish between the case of missing measures and the case of missing correspondences between nodes.

Synchronization over an inverse monoid

We observe that the notion of synchronization can be generalized to the case where Σ is an *inverse monoid*.

Definition 2.8. An *inverse semigroup* $(\Sigma, *)$ is a semigroup in which for all $s \in \Sigma$ there exists an element $t \in \Sigma$ such that $s = s * t * s$ and $t = t * s * t$. In this case, we write $t = s^{-1}$ and call t the *inverse* of s. If Σ has an identity element 1_Σ (i.e. it is a *monoid*), then it is called an *inverse monoid*.

Remark 2.10. Inverses in an inverse semigroup have many of the same properties as inverses in a group, for instance, $(a * b)^{-1} = b^{-1} * a^{-1}$ for all $a, b \in \Sigma$.

If Σ is an inverse monoid, then Equations (2.5) and (2.8) still make sense, with the provision that x_j^{-1} now denotes the inverse of x_j in the semigroup. Note that $x_j^{-1} * x_j$ and $x_j * x_j^{-1}$ are not necessarily equal to the identity, thus Equation (2.5) is not equivalent to (2.6). The solution to the synchronization problem over an inverse monoid is defined up to a global (right) product with any element $y \in \Sigma$ such that $y * y^{-1} = 1_\Sigma = y^{-1} * y$.

Spectral solution

Let $X_i \in ISym(d)$ denote the (unknown) label of vertex i and let $Z_{ij} \in ISym(d)$ denote the (known) label of edge $(i,j) \in \mathcal{E}$, which are linked by the consistency constraint $Z_{ij} = X_i X_j^\mathsf{T}$. If $[X_i]_{(h,k)} = 1$ for some index h we say that "node i sees object k".

Let X and Z be two block-matrices containing the vertex labels and edge labels respectively

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_n \end{bmatrix}, \quad Z = \begin{bmatrix} Z_{11} & Z_{12} & \dots & Z_{1n} \\ Z_{n1} & Z_{22} & \dots & Z_{2n} \\ \dots & & & \dots \\ Z_{n1} & Z_{n2} & \dots & Z_{nn} \end{bmatrix} \tag{2.90}$$

so that the consistency constraint becomes $Z = XX^\mathsf{T}$ with Z of rank d. Note that here the diagonal of Z is not filled with identity matrices, in contrast to Equation (2.44): since $X_i \in ISym(d)$, the $d \times d$ (diagonal) matrix $Z_{ii} = X_i X_i^\mathsf{T}$ is not equal, in general, to the identity, unless $X_i \in Sym(d)$. Indeed, $[Z_{ii}]_{(k,k)} = 1$ if node i sees object k and $[Z_{ii}]_{(k,k)} = 0$ otherwise. When all the objects seen by node i are different from those seen by node j we have $X_i X_j^\mathsf{T} = 0$, resulting in a zero block in Z.

Proposition 2.10. *The columns of X are d (orthogonal) eigenvectors of Z and the corresponding eigenvalues are contained in the diagonal of the following $d \times d$ matrix*

$$V := X^\mathsf{T} X = \sum_{i=1}^{n} X_i^\mathsf{T} X_i. \tag{2.91}$$

Proof. Using (2.91) and the consistency constraint, we obtain

$$ZX = XV \tag{2.92}$$

which is a spectral decomposition, i.e. the columns of X are d eigenvectors of Z and the corresponding eigenvalues are on the diagonal of V. Recall that Z admits an orthonormal basis of real eigenvectors since it is symmetric. □

Although $ISym(d)$ is not a group, an eigenvalue decomposition problem has been obtained, where the non-zero (and hence *leading*) eigenvalues are contained in the diagonal of V. Specifically, the k-th eigenvalue counts how many nodes see object k, thus all the eigenvalues are integer numbers lower than or equal to n. This implies that, when the number of objects is larger than the number of nodes (i.e., $d > n$) – which is likely to happen in the multi-view matching application – the eigenvalues are repeated. In the case of total permutations (i.e. $\Sigma = Sym(d)$) all the nodes see all the objects, thus $V = nI_d$ and all the eigenvalues are equal, hence Equation (2.92) reduces to (2.49).

In the presence of noise, the eigenvectors of Z corresponding to the d largest eigenvalues are computed, which solve Problem (2.73) as in the case of total permutations.

Equation (2.92) could also be expressed as a null-space problem. However, since V is unknown in practice, it should be estimated somehow.

Projection onto $ISym(d)$

Suppose that the eigenvectors of Z corresponding to the d largest eigenvalues are collected in a $nd \times d$ matrix U. Note that the reverse of Proposition 2.10 is not true in general, i.e., the matrix U is not necessarily equal to the vertex labelling X. Indeed, U is not uniquely determined if the eigenvalues of Z are repeated.

Proposition 2.11. *Let U be the $nd \times d$ matrix composed by the d leading eigenvectors of Z; then U has $d+1$ different rows (in the absence of noise). One of these is the zero row.*

Proof. Let $\lambda_1, \dots, \lambda_\ell$ denote all the *distinct* eigenvalues of Z (with $\ell \leq d$), and let $m_1, \dots, m_\ell$ be their multiplicities such that $\sum_{k=1}^{\ell} m_k = d$. Let U_{λ_k} denote the m_k columns of U corresponding to the eigenvalue λ_k, and let X_{λ_k} be the corresponding columns of X. Up to a permutation of the columns, we have

$$U = [U_{\lambda_1}\, U_{\lambda_2}\, \dots\, U_{\lambda_\ell}], \quad X = [X_{\lambda_1}\, X_{\lambda_2}\, \dots\, X_{\lambda_\ell}]. \tag{2.93}$$

Since U_{λ_k} and X_{λ_k} are (orthogonal) eigenvectors corresponding to the same eigenvalue, there exists an orthogonal matrix $Q_k \in \mathbb{R}^{m_k \times m_k}$ representing a change of basis in the eigenspace of λ_k, such that $U_{\lambda_k} = X_{\lambda_k} Q_k$. In matrix form this rewrites

$$U = X \underbrace{\operatorname{blkdiag}(Q_1, Q_2, \dots, Q_\ell)}_{Q}. \tag{2.94}$$

Note that the rows of X are the rows of I_d plus the zero row. Since Q is invertible (hence injective), $U = XQ$ has only $d+1$ different rows as well. □

Proposition 2.11 suggest that, in the presence of noise, an estimate of the vertex labelling can be obtained by clustering the rows of U into $d+1$ clusters (e.g. with k-means), then assigning the centroid which is closest to zero to the zero row, and arbitrarily assigning each of the other d centroids to a row of I_d. Recall that the solution to partial permutation synchronization is defined up to an element of $Sym(d)$. However, in this way, valid permutation matrices may not be obtained, as observed in the $\Sigma = Sym(d)$ case. For this reason, for each $d \times d$ block in U a partial permutation matrix that best maps such block into the set of centroids has to be computed (e.g. via the Kuhn-Munkres algorithm [110]).

2.11 Conclusion

In this chapter we gathered several disparate works within the common framework of synchronization, that is the problem of finding elements of a group given a certain number of their mutual differences (or ratios). Besides exhibiting a nice and clean formulation, synchronization can also benefit from efficient and closed-form solutions such as spectral decomposition or linear least squares. Chapter 4 will show how this framework can be profitably used in several Computer Vision applications. Future research will explore the link between the spectral solution and the consistency error of the synchronization problem when the data matrix is not symmetric. Besides this, we aim at analyzing in depth the synchronization problem over an inverse monoid, with particular focus to the notion of null-cycle, establishing how it relates to that of consistent labelling.

We also showed that the synchronization problem can be formulated as a low-rank and sparse matrix decomposition, that neatly caters for missing data, outliers and noise. This opens the way to the application of matrix decomposition techniques to several Computer Vision applications, since – in principle – any LRS algorithm can be plugged-in. In this respect, this approach will benefit from future developments in the field of LRS matrix decomposition.

Chapter 3

Bearing-based Localizability

This chapter provides a unifying view and offers new insights on bearing-based network localizability, that is the problem of establishing whether a set of directions between pairs of nodes uniquely determines (up to translation and scale) the position of the nodes in d-space. The contribution is theoretical: first, we rewrite and link in a coherent structure several results that have been presented in different communities using disparate formalisms; second, we derive some new localizability results within the edge-based formulation.

3.1 Introduction

Bearing-based (or *direction-based*) *network localization* is a fundamental problem in many computing and networking tasks. The goal is to recover the position of n nodes in d-space (with $n \geq 2$ and $d \geq 2$), given a redundant set of (possibly noisy) directions between pairs of nodes. Each node can represent any sensor able to measure the direction of the line joining its location to that of its neighbors (see [170, 136, 125]). The problem can be profitably modeled by introducing a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertices are the sensors and edges correspond to the available measures. Existing methods (e.g. [73, 33, 24, 162, 201]) differ in the problem formulation (deterministic versus probabilistic) and the computational model (centralized versus distributed).

A fundamental question concerns the *localizability* of the network – which is the main focus of this chapter, namely the problem of establishing whether bearing-based network localization is well-posed. Clearly, node locations can not be absolutely determined, since translations and dilations of a solution yield the same directions, and hence produce other solutions. Thus the question is whether the measures uniquely determine (up to translation and scale) the node locations, i.e., after fixing the position of two nodes (which essentially fixes the global translation and scale), all the other nodes are uniquely defined by the directions. Requiring that $\mathcal{G}$ is connected is not sufficient to guarantee the uniqueness of the solution, but more complicated assumptions are required, which are studied under the name of *parallel rigidity*.

Several theoretical results about parallel rigidity are present in the literature [191, 192, 50, 160, 5, 64, 62, 63, 146, 141, 201], as well as practical algorithms for finding maximal subgraphs in which the localization problem is well-posed [179, 103, 99].

These works come from disparate communities: discrete geometry [191, 192, 50, 99, 146]; computer vision [5, 141]; computer-aided design [160]; decision and control [64, 62, 63, 179, 201]; robotics [103].

The rigidity question can be posed either in terms of a *point formation* [64, 63], that is a configuration of n nodes in d-space, or in terms of the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ without reference to the specific values of the node locations [50], under the assumption that they are generic. The former gives rise to an algebraic characterization of localizability in terms of the rank of specific matrices derived from the coordinates of the nodes [62, 5, 141, 201], whereas the latter makes a combinatorial characterization of parallel rigidity possible [191, 192, 160]. A related concept is that of *parallel rigidity index* [146] in which the directions (instead of node locations) are assumed to be generic.

In this chapter we consider the *absolute* version of the localizability problem, namely we assume that all the directions are expressed in a common rotational reference frame. See [104] for the relative version of the problem, where no global coordinate frame is known. We also assume that the network is *anchor free*, i.e., all the nodes have unknown positions. Some localizability results in the presence of *anchors*, i.e. nodes whose position is known in advance, are reported in [201].

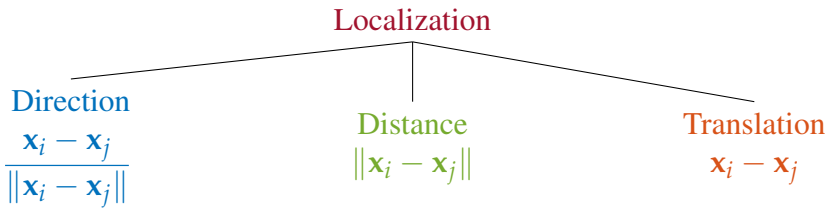


FIGURE 3.1: The localization problem. The goal is to compute the position $\mathbf{x}_i \in \mathbb{R}^d$ of n nodes given measures on the edges, which can be either directions (bearing-based localization), or distances (distance-based localization), or differences (translation synchronization). Only the latter is an instance of the synchronization problem.

A related topic, which is not covered in this chapter, is *distance-based network localization* [94, 25], where each node can measure relative distances to a set of other nodes. The corresponding theory on the uniqueness of the solution is known as *classical rigidity* [12], which characterizes well-posed instances of the localization problem in 2-space. However, no such characterization has been shown to hold for $d \geq 3$. Parallel rigidity, instead, has a simpler structure since the direction constraints between pairs of nodes can be expressed as linear equations, which allow to solve the localizability problem for all d.

If *both* distances and directions can be measured, then we are concerned with the problem of recovering the position of n nodes in d-space given a set of pairwise differences, which is a translation synchronization (see Section 2.3). Note that the bearing-based localization problem can not be viewed as a particular instance of synchronization since only directions are available, which are *not* differences of node locations. These different localization problems are illustrated in Figure 3.1.

3.1.1 Contribution

This chapter is conceived as a survey on bearing-based localizability, which summarizes the research that has been carried out in this area. Such research can be divided into two categories, according to the approach is used to address the localizability problem: node-based [191, 192, 50, 160, 64, 62, 63, 141, 103, 201] and edge-based [5, 179, 99, 146]. The former reasons in terms of node positions, whereas the latter reasons in terms of edge lengths. Note that the node-based formulation is better studied than the edge-based one, which is fairly recent.

In this chapter – for the first time in the literature – we provide a *unifying* view of bearing-based localizability, rewriting results proposed in different scenarios using the same theoretical formalism, while at the same time also proposing some novel results which fill in gaps of knowledge, particularly related to the edge-based formulation. More precisely, our contributions are the following.

First, we report in Section 3.2 an introduction to bearing-based network localization, where we show how such a problem can be reduced to a system of equations involving either node locations or edge lengths only. Although the localizability aspect is the focus of this chapter, deriving such equations may be useful to better understand the remaining sections.

Secondly, we provide a comprehensive survey on node-based parallel rigidity, which is reported in Section 3.3, considering both the standard definition of parallel rigidity, which involves a formation of n nodes in d-space, and the concept of generic parallel rigidity, which is a property of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A novel result linking rigidity in $\mathbb{R}^d$ with rigidity in $\mathbb{R}^{d+1}$ is also derived (Proposition 3.2).

Then, as our main contribution, in Section 3.4 we describe the edge-based formulation of parallel rigidity, which builds upon [99, 5, 179]. We show that this formulation is theoretically equivalent to the node-based one (Proposition 3.4), it entails a compact matrix formulation (Theorem 3.6), and it also enables us to prove results involving the structure of the graph. Specifically, we prove that biconnectivity is necessary for parallel rigidity (Proposition 3.7) and we derive sufficient conditions for parallel rigidity (Theorems 3.8 and 3.9) based on the existence of cycle bases of $\mathcal{G}$ with certain properties.

Finally, in Section 3.5 we rewrite in simpler terms the concept of parallel rigidity index defined in [146], providing its direct computation for some simple cases (Proposition 3.10 and Corollary 3.3) and explaining how it relates to parallel rigidity. We also underline its impact on bearing-based network localization, specifically we point out which graphs are better suited for the localization problem since they promote error compensation in the presence of noise.

3.2 Bearing-based Localization

Consider a network consisting of n nodes labelled from 1 to n, where each node is located at a fixed (unknown) position in $\mathbb{R}^d$, and suppose that some pairs of nodes

can measure the direction of the line joining their locations. Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the location of node i and let $\mathbf{u}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/\|\mathbf{x}_i - \mathbf{x}_j\|$ denote the direction between node i and node j. In some applications the sign of the directions may be unspecified, in which case $\pm\mathbf{u}_{ij}$ is known. The neighbor relationships of the network can be represented as a simple directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E}$ such that $(i, j) \in \mathcal{E}$ if there is a measured direction between node i and node j, with $m = |\mathcal{E}|$. We do not assume full measurements, i.e. $\mathcal{G}$ may not be complete.

The d-dimensional *bearing-based network localization problem* consists in determining the node locations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$, given the (redundant) set of pairwise measures $\{\mathbf{u}_{ij}\}_{(i,j)\in\mathcal{E}}$. These measures may be corrupted by noise, in which case the goal is to combine them into an estimate of the locations where errors get compensated, as shown in Figure 3.2. The remainder of this section gives an overview of two different approaches that solve the localization problem, assuming that it is well-posed.

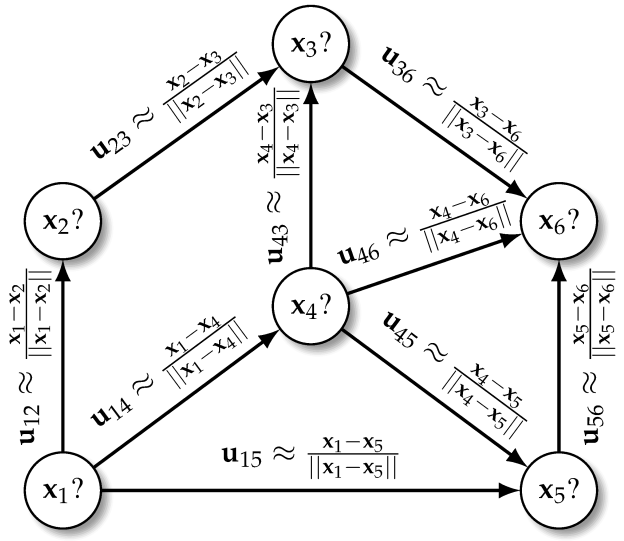


FIGURE 3.2: The bearing-based localization problem. Each node is located at an unknown position in space and measures on the edges are directions between pairs of nodes. The goal is to compute the positions that best agree with the measures.

Given a set of (noise-free) directions and the underlying graph, the unknown node locations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ can be recovered as the solution of the following system

$$\mathbf{x}_i - \mathbf{x}_j = \alpha_{ij}\mathbf{u}_{ij} \quad (i, j) \in \mathcal{E} \tag{3.1}$$

where $\alpha_{ij} \in \mathbb{R}$ are unknown as well. Note that, if the scales were known, this would reduce to a translation synchronization problem, namely Equation (2.17) with $\mathbf{z}_{ij} = \alpha_{ij}\mathbf{u}_{ij}$.

As we did in Section 2.3.2, let X be the $d \times n$ matrix obtained by juxtaposing all the node locations, namely $X = [\mathbf{x}_1 \ldots \mathbf{x}_n]$, let U be the $d \times m$ matrix obtained by juxtaposing all the edge directions, namely $U = [\mathbf{u}_{12} \ \ldots \ \mathbf{u}_{ij} \ \ldots]$, and let $\boldsymbol{\alpha} \in \mathbb{R}^m$

be the vector containing all the scales α_{ij}. It is easy to see that the equations above can be expressed in matrix form as

$$XB = U\,\mathrm{diag}(\boldsymbol{\alpha}) \tag{3.2}$$

where B denotes the $n \times m$ incidence matrix of $\mathcal{G}$. Applying the vectorization operator $\mathrm{vec}(\cdot)$ to both sides in (3.2) and using formulas (A.5) and (A.10) we get

$$(B^{\mathsf{T}} \otimes I_d)\mathbf{x} = (I_m \odot U)\boldsymbol{\alpha} \tag{3.3}$$

where $\mathbf{x} = \mathrm{vec}(X)$ and, as customary, $\otimes$ denotes the Kronecker product and $\odot$ denotes the Khatri-Rao product.

Note that Equation (3.3) contains both $\mathbf{x} \in \mathbb{R}^{dn}$ and $\boldsymbol{\alpha} \in \mathbb{R}^m$ as unknowns. There are two paths that can be followed in order to remove one unknown: node-based and edge-based. The former, which is reported in Section 3.2.1, derives a system of equations in terms of the node locations only, whereas the latter, which is described in Section 3.2.2, reduces (3.3) to a system of equations having the sole scales as unknowns. Alternatively, node locations and edge scales can be computed simultaneously, as done in [134, 181, 140].

3.2.1 Node-based Approach

We observe that the localization problem can be expressed in an alternative form which is equivalent to Equation (3.1). Let us start with the $d = 3$ case. The vector $\mathbf{u}_{ij}$ is the direction of $\mathbf{x}_i - \mathbf{x}_j$ if and only if their cross-product is zero, namely

$$\mathbf{u}_{ij} \times (\mathbf{x}_i - \mathbf{x}_j) = \mathbf{0} \quad (i,j) \in \mathcal{E} \tag{3.4}$$

or, equivalently,

$$[\mathbf{u}_{ij}]_\times (\mathbf{x}_i - \mathbf{x}_j) = \mathbf{0} \quad (i,j) \in \mathcal{E} \tag{3.5}$$

where $[\mathbf{a}]_\times$ denotes the skew-symmetric matrix associated to the cross-product with $\mathbf{a} = [a_1\ a_2\ a_3]^{\mathsf{T}}$, namely

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{3.6}$$

Equation (3.5) gives rise to 3 homogeneous equations for each edge in $\mathcal{E}$, where only two of them are linearly independent.

In the general case, the property that $\mathbf{u}_{ij}$ and $\mathbf{x}_i - \mathbf{x}_j$ have the same direction can be expressed as a rank constraint, namely

$$\mathrm{rank}([\mathbf{u}_{ij},\ \mathbf{x}_i - \mathbf{x}_j]) = 1 \quad (i,j) \in \mathcal{E} \tag{3.7}$$

which is equivalent to impose that all order-two minors are zero. The number of such minors is $d(d-1)/2$, which give rise to $d(d-1)/2$ homogeneous equations for each edge in $\mathcal{E}$, where only $d-1$ of them are linearly independent. Such equations can be expressed in matrix form as in Equation (3.5), where $[\mathbf{a}]_\times$ now denotes a $d(d-1)/2 \times d$ matrix composed of $d-1$ blocks arranged by rows, as explained in [124]. The i-th block has $d-i$ rows and d columns

$$A_i = \begin{bmatrix} \mathbf{0}_{1\times(i-1)} & -a_{i+1} & a_i & 0 & 0 & \dots & 0 \\ \mathbf{0}_{1\times(i-1)} & -a_{i+2} & 0 & a_i & 0 & \dots & 0 \\ \mathbf{0}_{1\times(i-1)} & -a_{i+3} & 0 & 0 & a_i & \dots & 0 \\ \dots & & & & & & \dots \\ \mathbf{0}_{1\times(i-1)} & -a_d & 0 & 0 & 0 & \dots & a_i \end{bmatrix} \tag{3.8}$$

and

$$[\mathbf{a}]_\times = \begin{bmatrix} A_1 \\ \dots \\ A_{d-1} \end{bmatrix}. \tag{3.9}$$

It can be checked that Equation (3.9) reduces to (3.6) if $d = 3$.

If we collect the equations in (3.5) for all the edges in $\mathcal{E}$, we get a system of $md(d-1)/2$ homogeneous equations in dn unknowns. Let S denote the $md(d-1)/2 \times md$ block-diagonal matrix with blocks $[\mathbf{u}_{ij}]_\times$ along the diagonal, namely

$$S = \text{blkdiag}(\{[\mathbf{u}_{ij}]_\times\}_{(i,j)\in\mathcal{E}}). \tag{3.10}$$

Using this notation, the equations in (3.5) can be expressed in a compact matrix form as

$$S \,\text{vec}(XB) = \mathbf{0} \tag{3.11}$$

which, using formula (A.5), rewrites

$$S(B^\mathsf{T} \otimes I_d)\mathbf{x} = \mathbf{0}. \tag{3.12}$$

Alternatively, the above equation can be derived by multiplying (3.3) by the block-diagonal matrix S

$$S(B^\mathsf{T} \otimes I_d)\mathbf{x} = S(I_m \odot U)\boldsymbol{\alpha}. \tag{3.13}$$

Note that, by construction, the right-side vanishes, yielding Equation (3.12).

Thus the unknown node locations are recovered as the solution of a homogeneous linear system which does not involve the edge scales, that is why this approach is called *node-based*. Theoretical conditions under which such solution is unique are studied in Section 3.3 under the name of parallel rigidity. This formulation is exploited in [78, 52] in the context of structure-from-motion.

A different formulation is used in [33, 141, 201, 76], which is based on the observation that $\mathbf{u}_{ij}$ is the direction of $\mathbf{x}_i - \mathbf{x}_j$ if and only if the components of $\mathbf{x}_i - \mathbf{x}_j$

orthogonal to $\mathbf{u}_{ij}$ are zero, namely

$$\Psi_{\mathbf{u}_{ij}}(\mathbf{x}_i - \mathbf{x}_j) = \mathbf{0} \quad (i,j) \in \mathcal{E} \tag{3.14}$$

where $\Psi_{\mathbf{a}} \in \mathbb{R}^{d\times d}$ denotes the orthogonal projection matrix which geometrically projects any vector onto the orthogonal compliment of $\mathbf{a} \in \mathbb{R}^d$, namely

$$\Psi_{\mathbf{a}} = I_d - \frac{\mathbf{a}\,\mathbf{a}^\mathsf{T}}{||\mathbf{a}||^2}. \tag{3.15}$$

Thus d homogeneous equations for each edge in $\mathcal{E}$ are obtained, where only $d-1$ are linearly independent. Such equations can be collected for all the edges as in Equation (3.12), resulting in

$$G(B^\mathsf{T} \otimes I_d)\mathbf{x} = \mathbf{0}. \tag{3.16}$$

where

$$G = \text{blkdiag}(\{\Psi_{\mathbf{u}_{ij}}\}_{(i,j)\in\mathcal{E}}) \tag{3.17}$$

resulting in a system of dm homogeneous equations in dn unknowns.

The bearing Laplacian matrix

In practice, the unknown node locations can be recovered by solving the normal equations associated to (3.16), namely

$$\underbrace{\big(G(B^\mathsf{T} \otimes I_d)\big)^\mathsf{T}\big(G(B^\mathsf{T} \otimes I_d)\big)}_{H}\mathbf{x} = \mathbf{0}. \tag{3.18}$$

Note that H is symmetric and positive semidefinite. By computation it can be verified that the projection matrix $\Psi_{\mathbf{a}}$ defined in (3.15) satisfies $\Psi_{\mathbf{a}}^\mathsf{T} = \Psi_{\mathbf{a}} = \Psi_{\mathbf{a}}^2$ for all $\mathbf{a} \in \mathbb{R}^d$, thus $G^\mathsf{T}G = G$. Combining this property with Equation (A.2) we get

$$H = (B \otimes I_d)G(B^\mathsf{T} \otimes I_d) \tag{3.19}$$

which, after some rewriting, can be expressed as

$$H = \begin{bmatrix} \sum_j \Psi_{\mathbf{u}_{1j}} & & & \\ & \sum_j \Psi_{\mathbf{u}_{2j}} & & \\ & & \ddots & \\ & & & \sum_j \Psi_{\mathbf{u}_{nj}} \end{bmatrix} - \begin{bmatrix} 0 & \Psi_{\mathbf{u}_{12}} & \dots & \Psi_{\mathbf{u}_{1n}} \\ \Psi_{\mathbf{u}_{21}} & 0 & \dots & \Psi_{\mathbf{u}_{2n}} \\ \vdots & & \ddots & \vdots \\ \Psi_{\mathbf{u}_{n1}} & \Psi_{\mathbf{u}_{n2}} & \dots & 0 \end{bmatrix}. \tag{3.20}$$

The matrix H is called the *bearing Laplacian* in [201] since it resembles the Laplacian matrix of a weighted graph. Note that Equation (3.20) is the same as the matrix used in [33].

Therefore

$$\min_{\|\mathbf{x}\|=1} \mathbf{x}^\mathsf{T} H \mathbf{x} = \min_{\|\mathbf{x}\|=1} \|G(B^\mathsf{T} \otimes I_d)\mathbf{x}\|^2 = \min_{\|\mathbf{x}\|=1} \sum_{(i,j)\in\mathcal{E}} \|\Psi_{\mathbf{u}_{ij}}(\mathbf{x}_i - \mathbf{x}_j)\|^2 \quad (3.21)$$

where the constraint $\|\mathbf{x}\| = 1$ fixes the global scale. This means that, in the presence of noise, the least-squares solution associated to Equation (3.14), where the components of the displacements that are orthogonal to the constraints are minimized, coincides with the spectral solution developed in [33], where the quadratic form associated to the bearing Laplacian is minimized.

Problem (3.21) admits a closed-form solution, which is given by the eigenvector of H associated to the smallest eigenvalue. However, trivial solutions exist, namely $\mathbf{x}_i = \mathbf{x}_j$ for all i, j, that correspond to mapping all the nodes to a single point in $\mathbb{R}^d$, which must be avoided. Note that this trivial subspace is spanned by the columns of $\mathbf{1}_n \otimes I_d \in \mathbb{R}^{dn\times d}$. Thus the sought solution must belong to $\ker(H)$ and be orthogonal to $\mathbf{1}_n \otimes I_d$ at the same time, in order to avoid trivial solutions. To compute it, it is sufficient to project H onto an orthogonal basis of $\ker(\mathbf{1}_n \otimes I_d)$, compute the eigenvalue decomposition of the reduced problem and then back project the eigenvectors, as explained in [33].

In the $d = 3$ case the bearing Laplacian can be equivalently expressed as

$$H = (B \otimes I_d)S^\mathsf{T} S(B^\mathsf{T} \otimes I_d) \quad (3.22)$$

where S is defined in (3.10), since $[\mathbf{a}]_\times[\mathbf{b}]_\times = \mathbf{b}\mathbf{a}^\mathsf{T} - \mathbf{a}\mathbf{b}^\mathsf{T} I_3$ for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, which implies that $[\mathbf{u}_{ij}]_\times^\mathsf{T}[\mathbf{u}_{ij}]_\times = I_3 - \mathbf{u}_{ij}\mathbf{u}_{ij}^\mathsf{T} = \Psi_{\mathbf{u}_{ij}}$. Note that the matrix in Equation (3.22) is the coefficient matrix of the normal equations associated to (3.12), thus

$$\min_{\|\mathbf{x}\|=1} \mathbf{x}^\mathsf{T} H \mathbf{x} = \min_{\|\mathbf{x}\|=1} \|S(B^\mathsf{T} \otimes I_3)\mathbf{x}\|^2 = \min_{\|\mathbf{x}\|=1} \sum_{(i,j)\in\mathcal{E}} \|\mathbf{u}_{ij} \times (\mathbf{x}_i - \mathbf{x}_j)\|^2 \quad (3.23)$$

which means that, in the presence of noise, the least-squares solution associated to (3.5) coincides with the spectral solution proposed in [33], as observed also in [103].

3.2.2 Edge-based Approach

Let us consider the cycle matrix $C \in \{-1, 0, 1\}^{(m-n+1)\times m}$ associated to a cycle basis of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and let us multiply left and right sides in (3.3) by $(C \otimes I_d)$

$$(C \otimes I_d)(B^\mathsf{T} \otimes I_d)\mathbf{x} = (C \otimes I_d)(I_m \odot U)\boldsymbol{\alpha}. \quad (3.24)$$

Using properties (A.4) and (A.9) we get

$$\big((CB^\mathsf{T}) \otimes I_d\big)\mathbf{x} = (C \odot U)\boldsymbol{\alpha}. \quad (3.25)$$

Since $CB^\mathsf{T} = 0$ for any cycle matrix C, as stated by Equation (B.13), a homogeneous linear system of equations in $\boldsymbol{\alpha}$ is obtained

$$\mathbf{0} = (C \odot U)\boldsymbol{\alpha}. \tag{3.26}$$

Note that Equation (3.26) is the null-cycle constraint of translation synchronization, namely it coincides with Equation (2.23) rewritten in terms of directions and scales.

We assume here that Equation (3.26) admits a unique solution (up to scale). Theoretical conditions under which such solution is unique are investigated in Section 3.4 and, as it will be shown, they are equivalent to parallel rigidity.

Once the unknown scales are recovered as the 1-dimensional null-space of $(C \odot U)$, the unknown node locations can be derived as the solution of Equation (3.3) with known $\boldsymbol{\alpha}$, which is a translation synchronization problem, thus its solution is unique (up to translation) if the graph is connected.

In summary, the unknown $\mathbf{x} \in \mathbb{R}^{dn}$ is recovered via a two-step method: first, the edge scales are computed as the solution of a homogeneous linear system, that is why this approach is called *edge-based*; then, the node locations are derived as the solution of a non-homogeneous linear system. This procedure will be exploited in Section 4.4 to recover camera location in a structure-from-motion pipeline. Experiments in Section 6.3 will reveal that this approach is comparable in accuracy to state-of-the-art algorithms which exploit the node-based formulation, thus showing that both node-based and edge-based algorithms are viable in practice.

3.3 Node-based Parallel Rigidity

The theory of parallel rigidity is concerned with the problem of establishing if there are enough direction constraints and they are distributed well enough to ensure that all the feasible solutions to bearing-based network localization differ by translation and scale. The *node-based formulation* of parallel rigidity – which is the classical way to study the solvability of the localization problem – reasons in terms of node positions, and it is based on the concept of point formation. A complete treatment of this subject can be found in [191, 192, 160, 64, 62, 63].

Definition 3.1. A d-dimensional *point formation* (or *embedding*) $\mathcal{F}_\mathbf{x}$ is a set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of n points in $\mathbb{R}^d$ together with a set $\mathcal{E}$ of m links, with $\mathcal{E} \subseteq \{(i,j), i \neq j, i, j \in \{1, 2, \dots, n\}\}$.

A point formation uniquely determines a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \{1, 2, \dots, n\}$ and edge set $\mathcal{E}$, together with a *measurement function* $\mathbf{u} : \mathcal{E} \to \mathbb{S}^{d-1}$ whose value at $(i,j) \in \mathcal{E}$ is the direction of $\mathbf{x}_i - \mathbf{x}_j$, namely

$$\mathbf{u}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||}. \tag{3.27}$$

We assume here that the points in $\mathcal{F}_\mathbf{x}$ are distinct, i.e. $\mathbf{x}_j \neq \mathbf{x}_i$ for all $i \neq j$, so that Equation (3.27) is well defined. We use the notation $\mathbf{x} \in \mathbb{R}^{dn}$ to denote the stack of the coordinates of the points in $\mathcal{F}_\mathbf{x}$, that is $\mathbf{x} = [\mathbf{x}_1^\mathsf{T}\ \mathbf{x}_2^\mathsf{T}\ \dots\ \mathbf{x}_n^\mathsf{T}]^\mathsf{T}$.

A point formation represents a configuration of n nodes in d-space. Specifically, the points $\mathbf{x}_i$ represent a solution to bearing-only localization with $\mathbf{u}_{ij}$ known, and the set $\mathcal{E}$ corresponds to node pairs for which the direction can be measured, which define constraints between specific nodes.

Definition 3.2. Two point formations $\mathcal{F}_\mathbf{x}$ and $\mathcal{F}_\mathbf{y}$ on the same graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are called *parallel point formations* (or *parallel drawings*) if $\mathbf{x}_i - \mathbf{x}_j$ is parallel to $\mathbf{y}_i - \mathbf{y}_j$ for all $(i, j) \in \mathcal{E}$.

Note that $\mathbf{x}_i - \mathbf{x}_j$ is parallel to $\mathbf{y}_i - \mathbf{y}_j$ if and only if there exists a scale $s_{ij} \in \mathbb{R}$ such that

$$\mathbf{y}_i - \mathbf{y}_j = s_{ij}(\mathbf{x}_i - \mathbf{x}_j) \quad (i, j) \in \mathcal{E}. \tag{3.28}$$

similarly to Equation (3.1).

The parallelism constraint for the edge $(i, j) \in \mathcal{E}$ can be expressed in an equivalent form which does not involve the unknown scale s_{ij}. Let us start with the $d = 2$ case. Using the operator $(\cdot)^\perp$ for turning a plane vector by $\pi/2$ counterclockwise, such constraint can be written as

$$(\mathbf{x}_i - \mathbf{x}_j)^\perp \cdot (\mathbf{y}_i - \mathbf{y}_j) = 0 \quad (i, j) \in \mathcal{E}. \tag{3.29}$$

Thus, given a point formation $\mathcal{F}_\mathbf{x}$ in 2-space, a parallel drawing $\mathcal{F}_\mathbf{y}$ solves a homogeneous equation for each edge in $\mathcal{E}$, resulting in a linear system of m equations in $2n$ unknowns. In the $d \geq 3$ case, Equation (3.29) generalizes to

$$\begin{aligned} (\mathbf{x}_i - \mathbf{x}_j)_{N_1}^\mathsf{T}(\mathbf{y}_i - \mathbf{y}_j) = 0 \quad (i, j) \in \mathcal{E} \\ (\mathbf{x}_i - \mathbf{x}_j)_{N_2}^\mathsf{T}(\mathbf{y}_i - \mathbf{y}_j) = 0 \quad (i, j) \in \mathcal{E} \\ \dots \\ (\mathbf{x}_i - \mathbf{x}_j)_{N_{d-1}}^\mathsf{T}(\mathbf{y}_i - \mathbf{y}_j) = 0 \quad (i, j) \in \mathcal{E} \end{aligned} \tag{3.30}$$

where $(\mathbf{x}_i - \mathbf{x}_j)_{N_k}$ for $k = 1, \dots, d-1$ are (linearly independent) vectors that span the subspace orthogonal to $\mathbf{x}_i - \mathbf{x}_j$. The equations in (3.30) are called the *direction constraints* (or *normal constraints*). Thus, given a point formation $\mathcal{F}_\mathbf{x}$, all parallel drawings solve $d-1$ homogeneous equations for each edge in $\mathcal{E}$. If we collect such equations for all the edges we get a system of $m(d-1)$ equations in dn unknowns

$$R_{\mathcal{F}_\mathbf{x}}\mathbf{y} = \mathbf{0} \tag{3.31}$$

where $\mathbf{y} = [\mathbf{y}_1^\mathsf{T}\ \mathbf{y}_2^\mathsf{T}\ \dots\ \mathbf{y}_n^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{dn}$ and $R_{\mathcal{F}_\mathbf{x}} \in \mathbb{R}^{m(d-1)\times dn}$ is called the *parallel rigidity matrix*.

Given a point formation $\mathcal{F}_\mathbf{x}$, *trivially parallel* point formations are translations and dilations of $\mathcal{F}_\mathbf{x}$ (including the parallel point formation in which all the points are

coincident). Note that also a negative scaling of $\mathcal{F}_\mathbf{x}$ is considered trivially parallel to the original point formation. All the others (if they exist) are non trivial. Figure 3.3 shows an example: Figure 3.3a represents a point formation in $\mathbb{R}^2$; Figures 3.3b and 3.3c are dilations of the point formation in Figure 3.3a (in particular, the former is an expansion and the latter is a contraction); Figure 3.3d shows a translation of the point formation in Figure 3.3a; Figure 3.3e reports a non-trivially parallel point formation, since it can not be obtained from the point formation in Figure 3.3a by translation or dilation, although all the corresponding edges are parallel to each other.

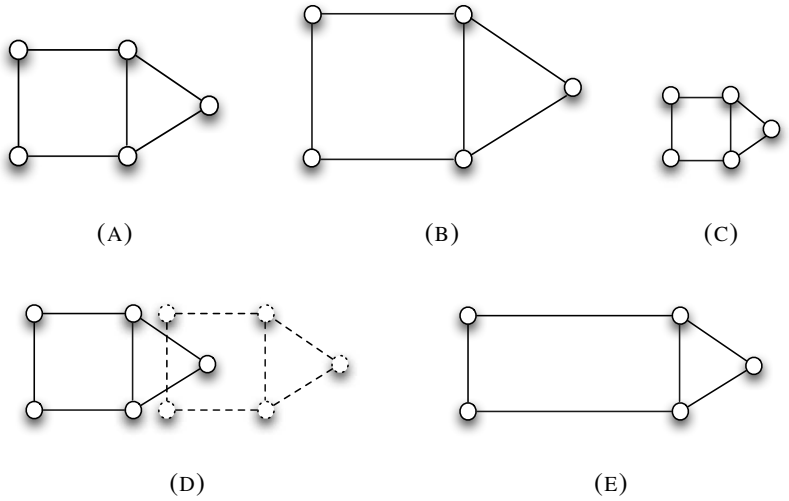


FIGURE 3.3: Parallel point formations.

Definition 3.3. A point formation $\mathcal{F}_\mathbf{x}$ is called *parallel rigid* (or *tight*) in d-space if all parallel point formations are trivially parallel. Otherwise it is called *flexible* (or *loose*).

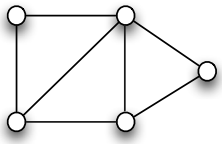


FIGURE 3.4: Parallel rigid point formation.

According to Definition 3.3, the point formation in Figure 3.3a is flexible in $\mathbb{R}^2$ since it admits a non-trivial parallel drawing, whereas the point formation in Figure 3.4, which is obtained from the formation in Figure 3.3a by adding an extra link, is parallel rigid. Note that parallel rigidity is a property of the point formation, i.e. it depends both on the graph and on the coordinates of the points in $\mathbb{R}^d$.

Note that trivially parallel point formations span a $(d+1)$-dimensional subspace of the null-space of $R_{\mathcal{F}_\mathbf{x}}$, corresponding to translations along each of d axes and one

scaling, thus $\text{rank}(R_{\mathcal{F}_\mathbf{x}}) \leq dn - (d+1)$. Since the set of parallel drawings of a point formation $\mathcal{F}_\mathbf{x}$ coincides with the solution set of Equation 3.31, we obtain the following algebraic characterization of parallel rigidity.

Theorem 3.1 ([62]). *Suppose that a point formation $\mathcal{F}_\mathbf{x}$ contains at least $d+1$ points which are not contained in any proper hyperplane of $\mathbb{R}^d$. $\mathcal{F}_\mathbf{x}$ is parallel rigid in d-space if and only if*

$$\dim(\ker(R_{\mathcal{F}_\mathbf{x}})) = d+1, \tag{3.32}$$

or, equivalently, if and only if

$$\text{rank}(R_{\mathcal{F}_\mathbf{x}}) = dn - (d+1). \tag{3.33}$$

Note that the assumption in Theorem 3.1 guarantees that there are always d independent solutions to (3.31). Otherwise, parallel rigidity can be checked in lower dimensions, namely in $(d-1)$-space if $\mathcal{F}_\mathbf{x}$ contains d points, and so on. In the remainder of this section we suppose that such assumption is satisfied.

Remark 3.1. We observe that a point formation $\mathcal{F}_\mathbf{x} = (\mathbf{x}, \mathcal{E})$ is parallel rigid if and only if $\mathcal{F}'_\mathbf{x} = (\mathbf{x}, \mathcal{E}')$ is parallel rigid, where $\mathcal{E}'$ is obtained from $\mathcal{E}$ by reversing the orientation of some edges, i.e. the endpoints are the same but the tail is replaced with the head and viceversa. Indeed, substituting (i,j) with (j,i) in Equation (3.30) results into an equivalent system. Thus the rigidity of a point formation is independent on the particular orientation of the edges, but it depends only on the underlying *undirected* graph. This is in agreement with Laman's condition (Theorem 3.3) which depends only on the number of edges/vertices of certain subgraphs of $\mathcal{G}$, but not on the orientation of the edges. For this reason, the figures of this chapter represent undirected graphs.

Remark 3.2. We observe that the directions constraints can be expressed in alternative forms which are equivalent to Equation (3.30). Following the same line as in Section 3.2.1, the vectors $\mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{y}_i - \mathbf{y}_j$ are parallel if and only if

$$\text{rank}([\mathbf{x}_i - \mathbf{x}_j,\ \mathbf{y}_i - \mathbf{y}_j]) = 1 \quad (i,j) \in \mathcal{E} \tag{3.34}$$

or, equivalently,

$$[\mathbf{x}_i - \mathbf{x}_j]_\times (\mathbf{y}_i - \mathbf{y}_j) = \mathbf{0} \quad (i,j) \in \mathcal{E}. \tag{3.35}$$

This formulation is used (e.g.) in [103, 179]. Using the Kronecker product $\otimes$ and the incidence matrix B, the equations in (3.35) can be collected for all the edges in $\mathcal{E}$, resulting in a system of $md(d-1)/2$ homogeneous equations in dn unknowns

$$S(B^\mathsf{T} \otimes I_d)\mathbf{y} = \mathbf{0} \tag{3.36}$$

where $S = \text{blkdiag}(\{[\mathbf{x}_i - \mathbf{x}_j]_\times\}_{(i,j)\in\mathcal{E}})$. The linear system in (3.36) is equivalent to (3.31), thus we can check the rank of $S(B^\mathsf{T} \otimes I_d)$ instead of the rank of the parallel

rigidity matrix to establish the rigidity of a point formation in d-space. Alternatively, $\mathbf{x}_i - \mathbf{x}_j$ is parallel to $\mathbf{y}_i - \mathbf{y}_j$ if and only if the components of $\mathbf{y}_i - \mathbf{y}_j$ orthogonal to $\mathbf{x}_i - \mathbf{x}_j$ are zero, namely

$$\Psi_{\mathbf{x}_i - \mathbf{x}_j}(\mathbf{y}_i - \mathbf{y}_j) = \mathbf{0} \quad (i,j) \in \mathcal{E}. \tag{3.37}$$

Such equations can be collected for all the edges in $\mathcal{E}$, resulting in a system of dm homogeneous equations in dn unknowns which is equivalent to (3.31), namely

$$G(B^\mathsf{T} \otimes I_d)\mathbf{y} = \mathbf{0} \tag{3.38}$$

where $G = \text{blkdiag}(\{\Psi_{\mathbf{x}_i - \mathbf{x}_j}\}_{(i,j)\in\mathcal{E}})$. Thus the rank of $G(B^\mathsf{T} \otimes I_d)$ can be computed to check the rigidity of a point formation, or equivalently, the rank of the bearing Laplacian matrix, as done in [201]. Recall that such a matrix is defined as $H = \big(G(B^\mathsf{T} \otimes I_d)\big)^\mathsf{T}\big(G(B^\mathsf{T} \otimes I_d)\big)$, thus $\text{rank}(H) = \text{rank}(G(B^\mathsf{T} \otimes I_d))$.

We introduce now the concept of global parallel rigidity which turns out to be equivalent to parallel rigidity. In simple words, a point formation is globally parallel rigid if it is the unique solution to the associated localization problem. In the case of distance-based localization, instead, the conditions for global rigidity are stronger than those for rigidity [12].

Definition 3.4. A point formation $\mathcal{F}_\mathbf{x}$ is called *globally parallel rigid* in d-space if it is exactly determined (up to translation and scale) by its graph and measurement function.

Proposition 3.1 ([63])**.** *A point formation $\mathcal{F}_\mathbf{x}$ is parallel rigid if and only if it is globally parallel rigid.*

Proof. We follow the reasoning reported in [139]. Let us consider the measurement function which associates a point formation $\mathcal{F}_\mathbf{x}$ with $\Psi_{\mathbf{x}_i - \mathbf{x}_j}$ for all $(i,j) \in \mathcal{E}$, where the orthogonal projection matrix Ψ is defined in (3.15). Note that the knowledge of $\Psi_{\mathbf{x}_i - \mathbf{x}_j}$ is equivalent to the knowledge of $\pm\mathbf{u}_{ij}$ where the sign is undetermined, with $\mathbf{u}_{ij}$ defined in (3.27). Let us define a map $\mathcal{M}$ that assigns each point formation to its graph and measurement function, namely

$$\mathcal{M} : \mathcal{F}_\mathbf{x} \mapsto (\mathcal{G}, \Psi). \tag{3.39}$$

More precisely, since being trivially parallel point formation is an equivalence relation, we can define the above map on the quotient space induced by such relation. Note that $\mathcal{F}_\mathbf{x}$ is globally parallel rigid if and only if $\mathcal{M}$ is injective at $\mathcal{F}_\mathbf{x}$. It is clear that $\mathcal{M}$ is injective at $\mathcal{F}_\mathbf{x}$ if and only if $\mathcal{F}_\mathbf{x}$ admits only trivially parallel point formations, since parallel point formations have the same measurement function and vice-versa. □

Remark 3.3. Proposition 3.1 implies that, in order to check the rigidity of a point formation, the equations coming from the localization problem – discussed in Section

3.2.1 – could be used instead of those arising from the definition of parallel rigidity, e.g. Equation (3.5) in place of Equation (3.35), or Equation (3.14) in place of Equation (3.37).

According to the above results, *given* a point formation $\mathcal{F}_\mathbf{x}$, it can be established (by checking the rank of the parallel rigidity matrix) if it is uniquely determined by its graph and directions. If it is not so, the cause can be the structure of the graph or the actual coordinates of the points, i.e., there can be algebraic dependencies among the coordinates that make the rank drop. How can we predict the rigidity of the problem based on the structure of the graph (and dimension d) only? This issue is addressed in Section 3.3.1, where the concept of generic parallel rigidity is introduced.

3.3.1 Generic Rigidity

We introduce the property of generic rigidity, which does not depend on the specific coordinates of a point formation, but it predicts the rigidity of almost all the point formations from the nodes and their incidences, i.e., from the underlying graph.

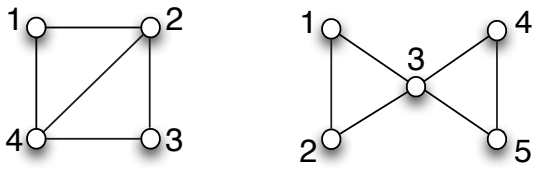
Definition 3.5. A set $\mathcal{A} = \{\alpha_1, \dots, \alpha_k\}$ of distinct real numbers is called *algebraically dependent* if there exists a non-zero polynomial h with integer coefficients such that $h(\alpha_1, \dots, \alpha_k) = 0$. Otherwise it is called *generic*.

A set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of points in $\mathbb{R}^d$ is called *generic* if its dn coordinates are generic. Note that if the set contains less than $d+1$ points, then there are always algebraic dependencies among the coordinates, since the points are contained in an hyperplane of $\mathbb{R}^d$. In such a situation the generic property is checked in lower dimensions (namely $d-1$ if there are d points, and so on). It can be shown that the set of generic $\mathcal{X}$s forms an open dense subset of $\mathbb{R}^{dn}$ [50]. Note that, with reference to the structure-from-motion application, a camera that is moving along a straight line is not considered generic by this theory (if the points in the scene are not considered).

Definition 3.6. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called *generically parallel rigid* in d-space if $\mathcal{F}_\mathbf{x} = (\mathcal{X}, \mathcal{E})$ is parallel rigid for a generic $\mathcal{X}$. Otherwise it is called *generically flexible*.

Due to Theorem 3.1, we can equivalently say that a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid if and only if $\operatorname{rank}(R_{\mathcal{F}_\mathbf{x}}) = dn - (d+1)$, where $R_{\mathcal{F}_\mathbf{x}}$ is constructed using a generic point formation $\mathcal{F}_\mathbf{x}$. It can be shown that $\operatorname{rank}(R_{\mathcal{F}_\mathbf{x}})$ is independent on the coordinates of the chosen point formation (assuming that it is generic) and it depends only on the underlying graph and dimension d [62], hence Definition 3.6 is well-posed. Due to Proposition 3.1, we can equivalently say that a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid if $\mathcal{F}_\mathbf{x} = (\mathcal{X}, \mathcal{E})$ is *globally* parallel rigid for a generic $\mathcal{X}$.

Figure 3.5a shows a generically parallel rigid graph[1]. It is easy to see that, given a point formation with generic coordinates, all the possible transformations that keep the corresponding edges parallel are translations and dilations, thus the underlying graph is parallel rigid in d-space. Note that the rigidity of $\mathcal{G}$ does not imply that *all* the point formations defined on $\mathcal{G}$ are parallel rigid. If there are algebraic dependencies among the coordinates (e.g. all the nodes lie on a common line) then the rank of the parallel rigidity matrix drop, resulting in a flexible point formation. The rigidity of $\mathcal{G}$ implies that all the *generic* point formations are parallel rigid, and hence, since generic formations are dense in $\mathbb{R}^{dn}$, it implies that *almost all* the formations defined on $\mathcal{G}$ are parallel rigid in d-space. Figure 3.5b shows a generically flexible graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. It is easy to see that, given a point formation with generic coordinates, independent scaling of the edges in the subgraphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1) = (\{1,2,3\}, \{(1,2),(2,3),(3,1)\})$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2) = (\{3,4,5\}, \{(3,4),(4,5),(5,3)\})$ produces non-trivially parallel point formations, thus the underlying graph is flexible in d-space.



(A) (B)

FIGURE 3.5: Left: generically parallel rigid graph. Right: generically flexible graph.

Note that, if we choose a graph and vary the position of the nodes in d-space, the dimension of the solution space of Equation (3.31) changes. However, a minimum dimension occurs and it clearly depends only on the underlying graph. It can be shown that such minimum is attained when the coordinates of the nodes are generic [62], which is equivalent to say that generic point formations maximize the rank of the parallel rigidity matrix. Recall that $\text{rank}(R_{\mathcal{F}_\mathbf{x}})$ is constant for any generic $\mathcal{F}_\mathbf{x}$. Thus we get the following algebraic characterization of generic parallel rigidity.

Theorem 3.2 ([62]). *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if*

$$\max_{\mathbf{x} \in \mathbb{R}^{nd}} \text{rank}(R_{\mathcal{F}_\mathbf{x}}) = dn - (d+1). \tag{3.40}$$

Theorem 3.2 gives rise to a randomized test for checking generic parallel rigidity [141], where a parallel rigidity matrix is built from a point formation randomly sampled from i.i.d. Gaussian distribution. This test correctly establishes the generic rigidity of $\mathcal{G}$ with probability 1 with a time complexity of $O(m)$, with $m = |\mathcal{E}|$.

[1] Note that Figures 3.3 and 3.4 represent 2-dimensional point formations, i.e. specific coordinates of nodes in 2-space, whereas Figure 3.5 and all subsequent figures represent graphs, i.e. only the links matter and the embedding in the plane is merely accidental.

Note that, if $\mathcal{G}$ is generically parallel rigid, then adding edges between existing nodes keeps the graph rigid, since it corresponds to include dependent equations in (3.31). If removing an edge results into a flexible graph, then $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called *minimally parallel rigid* in d-space, i.e. it is generically parallel rigid with minimum number of constraints. See [64] for techniques to generate minimally rigid graphs in 2-space and in 3-space.

We now list some combinatorial characterizations of generic parallel rigidity.

Theorem 3.3 (Laman's condition [192])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if there exists a subset $\mathcal{E}' \subseteq (d-1)\mathcal{E}$, where $(d-1)\mathcal{E}$ denotes the set consisting of $d-1$ copies of the edges in $\mathcal{E}$, such that the following conditions are satisfied:*

1. $|\mathcal{E}'| = dn - (d+1)$;
2. $\forall\, \mathcal{E}'' \subseteq \mathcal{E}'$, $\mathcal{E}'' \neq \varnothing$: $|\mathcal{E}''| \leq d|\mathcal{V}''| - (d+1)$, *where $\mathcal{V}''$ denotes the set of vertices that are endpoints of the edges in $\mathcal{E}''$.*

Corollary 3.1 ([191])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if for any partition $\{\mathcal{E}^1, \mathcal{E}^2, \ldots, \mathcal{E}^h\}$ of $\mathcal{E}$ it holds*

$$\sum_{i=1}^{h} \big(d|\mathcal{V}^i| - (d+1)\big) \geq dn - (d+1). \tag{3.41}$$

The conditions in Theorem 3.3 translate into combinatorial algorithms for testing generic parallel rigidity, e.g. methods based on the pebble game [93] with a time complexity of $O(n^2)$. Note that if $\mathcal{E}$ satisfies $(d-1)|\mathcal{E}| < dn - (d+1)$ then, even if we take $\mathcal{E}' = (d-1)\mathcal{E}$, Condition 1 in Theorem 3.3 can not be fulfilled. Thus we have the following *necessary* condition for a graph $\mathcal{G}$ to be generically parallel rigid

$$(d-1)m \geq dn - (d+1) \tag{3.42}$$

which essentially states that $\mathcal{G}$ needs to have a sufficient number of edges.

Let us consider the examples provided in Figure 3.5. The graph in Figure 3.5a is generically parallel rigid in 2-space, since $\mathcal{E}' = \mathcal{E}$ satisfies the conditions in Theorem 3.3, whereas the graph in Figure 3.5b is generically flexible in 2-space, since the necessary condition (3.42) is not satisfied. Note that in the $d = 2$ case the set $\mathcal{E}'$ satisfying the conditions in Theorem 3.3 (if it exists) is simply a subset of the edge set $\mathcal{E}$. We now consider the $d = 3$ case. As for Figure 3.5a, it is easy to see that $\mathcal{E}' = \mathcal{E} \cup \{(1,2),(2,4),(4,1)\} \subseteq 2\mathcal{E}$ satisfies the conditions in Theorem 3.3. As for Figure 3.5b, since $|2\mathcal{E}| = 12$, there exists a set $\mathcal{E}'$ satisfying Condition 1, i.e. $|\mathcal{E}'| = 11$, only if $\mathcal{E}' = 2\mathcal{E} \setminus \{e\}$ for some $e \in \mathcal{E}$. However, in this case $\mathcal{E}'$ has a subset $\mathcal{E}''$ consisting of two copies of each of the three edges in a triangle graph, which violates Condition 2.

The following result derives from the equivalence between the count in Theorem 3.3 and a $(d+1)Td$ decomposition of $\mathcal{E}'$ [84], i.e. a decomposition of $\mathcal{E}'$ into $d+1$ edge-disjoint trees where each vertex is contained in d trees.

Theorem 3.4 ([192]). *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if there exists a subset $\mathcal{E}' \subseteq (d-1)\mathcal{E}$ such that $\mathcal{E}'$ can be decomposed into $d+1$ edge-disjoint trees, where each vertex is contained in exactly d trees, and for any subgraph $\mathcal{E}'' \subseteq \mathcal{E}'$, $\mathcal{E}'' \neq \varnothing$, the set of trees induced by $\mathcal{E}''$ has cardinality at least $d+1$.*

Note that Theorems 3.3 and 3.4 do not involve the whole graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, but the existence of a subset $\mathcal{E}' \subseteq (d-1)\mathcal{E}$ with certain properties. As observed in [191, 192], such subset corresponds to $dn-(d+1)$ linearly independent rows of the parallel rigidity matrix, whose existence is equivalent to $\operatorname{rank}(R_{\mathcal{F}_\mathbf{x}}) = dn-(d+1)$.

The following novel result, which exploits Laman's condition, establishes the relation between generic rigidity in d-space and in $(d+1)$-space. In particular, we get that all the graphs which are generically parallel rigid in $\mathbb{R}^2$ are also rigid in $\mathbb{R}^d$ for any $d \geq 3$.

Proposition 3.2. *If a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space, then it is generically parallel rigid in $(d+1)$-space.*

Proof. Since $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space, by virtue of Theorem 3.3, there exists a subset $\mathcal{E}' \subseteq (d-1)\mathcal{E}$ such that

$$|\mathcal{E}'| = dn-(d+1) \tag{3.43}$$

$$\forall\, \mathcal{E}'' \subseteq \mathcal{E}',\ \mathcal{E}'' \neq \varnothing:\ \ |\mathcal{E}''| \leq d|\mathcal{V}''| - (d+1). \tag{3.44}$$

Note that $\mathcal{G}$ is connected, otherwise a non-trivial parallel drawing can be found from independent scaling and/or translation of each connected component, which contradicts the assumption. In order to prove that $\mathcal{G}$ is generically parallel rigid in $(d+1)$-space, we have to find a set $\widetilde{\mathcal{E}}' \subseteq d\mathcal{E}$ satisfying the following conditions

$$|\widetilde{\mathcal{E}}'| = (d+1)n-(d+2) \tag{3.45}$$

$$\forall\, \widetilde{\mathcal{E}}'' \subseteq \widetilde{\mathcal{E}}',\ \widetilde{\mathcal{E}}'' \neq \varnothing:\ \ |\widetilde{\mathcal{E}}''| \leq (d+1)|\widetilde{\mathcal{V}}''| - (d+2). \tag{3.46}$$

Let us define $\widetilde{\mathcal{E}}' := \mathcal{E}' \cup \mathcal{T}$, where $\mathcal{T}$ is any arbitrary spanning tree of $\mathcal{G}$, which is well defined since $\mathcal{G}$ is connected. Note that $\mathcal{E}'$ is contained in the set consisting of $d-1$ copies of the edges of $\mathcal{E}$, whereas $\mathcal{T}$ is contained in $\mathcal{E}$, thus their union is contained in the set consisting of d copies of the edges of $\mathcal{E}$. Using Equation (3.43) and $|\mathcal{T}| = n-1$ we get $|\widetilde{\mathcal{E}}'| = |\mathcal{E}'| + |\mathcal{T}| = dn-(d+1)+(n-1) = (d+1)n-(d+2)$ and hence Equation (3.45) is satisfied. We now prove that Equation (3.46) holds. Let $\widetilde{\mathcal{E}}'' \subseteq \widetilde{\mathcal{E}}'$ with $\widetilde{\mathcal{E}}'' \neq \varnothing$. We can write $\widetilde{\mathcal{E}}'' = \mathcal{E}'' \cup \mathcal{T}''$ where $\mathcal{E}'' \subseteq \mathcal{E}'$ and $\mathcal{T}'' \subseteq \mathcal{T}$. Note that $\mathcal{T}''$ is not necessarily a tree, but it will be a disjoint union of trees (i.e. a forest) in general, thus $|\mathcal{T}''| = n_{\mathcal{T}''} - cc \leq n_{\mathcal{T}''} - 1$, where

cc denotes the number of connected components in $\mathcal{T}''$ and $n_{\mathcal{T}''}$ denotes the number of vertices that are endpoints of the edges in $\mathcal{T}''$. Combining this observation with Equation (3.44) we get $|\widetilde{\mathcal{E}}''| = |\mathcal{E}''| + |\mathcal{T}''| \leq d|\mathcal{V}''| - (d+1) + n_{\mathcal{T}''} - 1$. Note that the number of vertices that are endpoints of the edges in $\mathcal{E}''$ and the number of vertices that are endpoints of the edges in $\mathcal{T}''$ are both dominated by the total number of vertices in $\widetilde{\mathcal{E}}''$, i.e. $|\mathcal{V}''| \leq |\widetilde{\mathcal{V}}''|$ and $n_{\mathcal{T}''} \leq |\widetilde{\mathcal{V}}''|$, hence we get Equation (3.46). □

Note that the converse of Proposition 3.2 is not true. For instance, the graph associated to the point formation in Figure 3.3 is flexible in $\mathbb{R}^2$ and it is parallel rigid in $\mathbb{R}^d$ for any $d \geq 3$, as it can be easily checked.

Maximal rigid components.

If a graph is not generically parallel rigid, then it can be decomposed into *maximal rigid components*. A *rigid component* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in d-space is a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}'$ is generically parallel rigid in d-space. Clearly, the union of rigid components sharing (at least) one edge is also rigid, since the edge in common fixes the position of two nodes and hence it determines the global scale and translation. A rigid component is called *maximal* if it is not a subset of any other rigid component.

Theorem 3.5 ([103])**.** *The set of all maximal rigid components of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in d-space induces a partition of the edge set $\mathcal{E}$.*

For instance, the edge set of the (flexible) graph reported in Figure 3.5b can be partitioned into two maximal rigid components, namely $\mathcal{E}^1 = \{(1,2),(2,3),(3,1)\}$ and $\mathcal{E}^2 = \{(3,4),(4,5),(5,3)\}$.

Polynomial-time algorithms for finding maximal rigid components of flexible graphs are presented in [103, 99]. The authors of [103] analyze the null-space of the parallel rigidity matrix and cast the problem to identifying sets of parallel lines. A different approach is followed in [99] where rigid components are first identified among known rigid graphs of small size, and then they are grouped using a reduction to a maximum flow problem.

3.4 Edge-based Parallel Rigidity

In this section we describe an equivalent formulation of parallel rigidity, which is called the *edge-based formulation*, since it reasons in terms of edge lengths rather than node positions. It provides a more intuitive way to look at rigidity, since the problem is expressed in terms of cycles in the graph. This formulation is based on some recent works [99, 5, 179].

Let $\mathcal{F}_{\mathbf{x}}$ be a point formation in d-space and let $\alpha_{ij} \in \mathbb{R}^+$ denote the *length* of $\mathbf{x}_i - \mathbf{x}_j$ for $(i,j) \in \mathcal{E}$, namely

$$\alpha_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||. \tag{3.47}$$

The inter-nodal distances α_{ij} are called the *edge scales* in [179], and they are also known as *epipolar scales*, with reference to the structure-from-motion application. Alternatively, given a point formation $\mathcal{F}_\mathbf{x}$ and its measurement function, i.e. its set of directions $\{\mathbf{u}_{ij}\}$, we can define the length of edge (i,j) as the positive real number α_{ij} such that Equation (3.1) holds, i.e. $\mathbf{x}_i - \mathbf{x}_j = \alpha_{ij}\mathbf{u}_{ij}$. This general definition can also take into account the fact that a direction $\mathbf{u}_{ij}$ may be measured with the wrong sign, in which case the corresponding α_{ij} is negative in order to fulfill Equation (3.1).

We now show how parallel rigidity can be restated in terms of edge lengths.

Proposition 3.3 ([179])**.** *A point formation $\mathcal{F}_\mathbf{x}$ is parallel rigid if and only if for any parallel point formation $\mathcal{F}_\mathbf{y}$ it holds*

$$\mathbf{y}_i - \mathbf{y}_j = s(\mathbf{x}_i - \mathbf{x}_j) \quad \forall\, (i,j) \in \mathcal{E} \tag{3.48}$$

assuming that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected, where s does not depend on the individual (i,j) pair.

Proof. In one direction: if $\mathcal{F}_\mathbf{x}$ is parallel rigid then – by definition – any parallel drawing $\mathcal{F}_\mathbf{y}$ satisfies $\mathbf{y}_j = s\mathbf{x}_j + \mathbf{t}$ with $s \in \mathbb{R}$ and $\mathbf{t} \in \mathbb{R}^d$, and hence Equation (3.48) clearly holds. In the opposite direction: let $\mathcal{F}_\mathbf{x}$ be a point formation and let $\mathcal{F}_\mathbf{y}$ be a parallel drawing such that $\mathbf{y}_i - \mathbf{y}_j = s(\mathbf{x}_i - \mathbf{x}_j)$ for all $(i,j) \in \mathcal{E}$, or, equivalently, $\mathbf{y}_i - s\mathbf{x}_i = \mathbf{y}_j - s\mathbf{x}_j$. If the graph is connected, each node can be reached by any other node through a path, thus such relation is valid for all the nodes, i.e. $\mathbf{y}_i - s\mathbf{x}_i = \mathbf{y}_j - s\mathbf{x}_j = \cdots = \mathbf{y}_k - s\mathbf{x}_k = \mathbf{t}$, thus $\mathbf{y}_j = s\mathbf{x}_j + \mathbf{t}$ for all $i \in \mathcal{V}$, which means that $\mathcal{F}_\mathbf{x}$ is parallel rigid. □

A set of edges satisfying Equation (3.48) is called *interdependent edge set* in [179]. Note that, according to Equation (3.28), the parallelism of $\mathcal{F}_\mathbf{x}$ and $\mathcal{F}_\mathbf{y}$ rewrites $\mathbf{y}_i - \mathbf{y}_j = s_{ij}(\mathbf{x}_i - \mathbf{x}_j)$ for some scales $s_{ij} \in \mathbb{R}$, while parallel rigidity – as expressed by Equation (3.48) – means that such scales are all equal, i.e. $s_{ij} = s \in \mathbb{R}$ for all $(i,j) \in \mathcal{E}$.

Proposition 3.4. *A point formation $\mathcal{F}_\mathbf{x}$ is parallel rigid if and only if its lengths are exactly determined (up to a global scale) by its graph and measurement function, assuming that the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected.*

Proof. In one direction: if $\mathcal{F}_\mathbf{x}$ is parallel rigid then, due to (3.48), any parallel drawing $\mathcal{F}_\mathbf{y}$ satisfies $||\mathbf{y}_i - \mathbf{y}_j|| = |s| \cdot ||\mathbf{x}_i - \mathbf{x}_j||$ for all $(i,j) \in \mathcal{E}$, i.e. the lengths of $\mathcal{F}_\mathbf{y}$ coincide (up to a global scale) with those of $\mathcal{F}_\mathbf{x}$. Recall that parallel point formations have the same measurement function and vice-versa, hence we get the thesis. In the opposite direction: by definition, the lengths of $\mathcal{F}_\mathbf{x}$ satisfy (3.1), which is equivalent to Equation (3.3). As explained in Section 3.2.2, if the graph is connected, then Equation (3.3) admits a unique solution (up to translation) for fixed lengths, i.e. $\mathcal{F}_\mathbf{x}$ is uniquely determined (up to translation) by its graph and measurement function. Combining this observation with the assumption, we get that $\mathcal{F}_\mathbf{x}$ is uniquely determined (up to translation and scale) by its graph and directions, i.e. it is globally parallel rigid (and hence parallel rigid). □

In simple words, Proposition 3.4 states that we cannot change the inter-nodal distances of a parallel-rigid point formation independently since, by fixing the length of an edge, we also constrain the length of the remaining edges.

We aim at deriving an algebraic characterization of parallel rigidity in terms of edge lengths, thus a linear system having the sole lengths as unknowns is required. Such system is reported in the following proposition and it involves suitable circuits in $\mathcal{G}$.

Proposition 3.5 ([99]). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph and let $\mathcal{T}$ be a spanning tree of $\mathcal{G}$. For any $e \in \mathcal{E} \setminus \mathcal{T}$ let $\mathbf{c}^e \in \{-1, 0, 1\}^m$ denote the circuit obtained by adding e to $\mathcal{T}$, and let $\mathbf{c}^e_+$ ($\mathbf{c}^e_-$) denote the forward (backward) edges in $\mathbf{c}_e$. Let $\{\mathbf{u}_{ij}\}$ be a set of directions defined on $\mathcal{G}$. A length assignment $\{\alpha_{ij}\}$ is compatible with edge directions $\{\mathbf{u}_{ij}\}$, i.e. there exists a point formation $\mathcal{F}_\mathbf{x}$ on $\mathcal{G}$ with directions $\mathbf{u}_{ij}$ and lengths α_{ij}, if and only if*

$$\sum_{(i,j)\in\mathbf{c}^e_+} \alpha_{ij}\mathbf{u}_{ij} - \sum_{(i,j)\in\mathbf{c}^e_-} \alpha_{ij}\mathbf{u}_{ij} = \mathbf{0} \quad \forall\, e \in \mathcal{E} \setminus \mathcal{T}. \tag{3.49}$$

Proof. Note that Equation (3.49) can be written as

$$\sum_{(i,j)\in\mathcal{E}} [\mathbf{c}^e]_{ij}\alpha_{ij}\mathbf{u}_{ij} = \mathbf{0} \quad \forall\, e \in \mathcal{E} \setminus \mathcal{T} \tag{3.50}$$

where the circuit $\mathbf{c}^e$ is traversed in a cyclic order (clockwise or anti-clockwise), and the (non-zero) entries of $\mathbf{c}^e$ have a sign that indicates whether the corresponding edge is traversed along the direction specified by $\mathbf{u}_{ij}$ or not. Equation (3.50) clearly holds if $\alpha_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ and $\mathbf{u}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/||\mathbf{x}_i - \mathbf{x}_j||$ for a point formation $\mathcal{F}_\mathbf{x}$. To prove the opposite direction, we can compute the position of the nodes using the spanning tree $\mathcal{T}$, i.e. the root is set equal to the zero vector and the coordinates of the other nodes are computed via the relation $\mathbf{x}_i = \mathbf{x}_j + \alpha_{ij}\mathbf{u}_{ij} \Leftrightarrow \mathbf{x}_i - \mathbf{x}_j = \alpha_{ij}\mathbf{u}_{ij}$. Such point formation has directions equal to $\mathbf{u}_{ij}$ and lengths equal to α_{ij} for all the edges $e \in \mathcal{T}$ (by construction), and also for all the edges $e \in \mathcal{E} \setminus \mathcal{T}$ (due to Equation (3.50)). □

Note that Proposition 3.5 is about the existence and not the uniqueness of a point formation. For instance, if $\mathcal{G}$ is a tree (which does not contain circuits) and $\{\mathbf{u}_{ij}\}$ is a given set of directions, then *any* length assignment is valid, whereas any edge beyond the tree introduces additional constraints.

The equations in (3.50) state that the (signed) sum of directions (weighted with the correct lengths) along circuits must be zero, which is exactly the null-cycle property for translation synchronization, rewritten in terms of directions and lengths. Such equations can be expressed in a compact matrix form if all the lengths α_{ij} are collected in a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ and all the directions $\mathbf{u}_{ij}$ are collected in a matrix

$U \in \mathbb{R}^{d\times m}$. Specifically, the equations for a single circuit $\mathbf{c}_e$ become

$$U\mathrm{diag}(\mathbf{c}_e^\mathsf{T})\boldsymbol{\alpha} = \mathbf{0} \tag{3.51}$$

or, equivalently, using the Khatri-Rao product

$$(\mathbf{c}_e^\mathsf{T} \odot U)\boldsymbol{\alpha} = \mathbf{0}. \tag{3.52}$$

If the equations coming from all the circuits induced by $\mathcal{T}$ are stacked, then a system of $d(m-n+1)$ homogeneous equations is obtained, namely

$$(C_\mathcal{T} \odot U)\boldsymbol{\alpha} = \mathbf{0} \tag{3.53}$$

where $C_\mathcal{T}$ denotes the cycle matrix associated to the circuits $\mathbf{c}_e$ for $e \in \mathcal{E} \setminus \mathcal{T}$, which indeed form a fundamental cycle basis.

The following result states that we can use *any* cycle basis (fundamental or not) in Equation (3.53).

Proposition 3.6. *Equation* (3.53) *is equivalent to*

$$(C \odot U)\boldsymbol{\alpha} = \mathbf{0} \tag{3.54}$$

where C denotes the cycle matrix associated to any cycle basis of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Proof. Let $C_\mathcal{T}$ be the cycle matrix associated to the circuits defined in Proposition 3.5 and let C be the cycle matrix associated to another cycle basis of $\mathcal{G}$. Since the cycle space of the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a vector space over $\mathbb{Q}$, there exists an invertible matrix $R \in \mathbb{Q}^{(m-n+1)\times(m-n+1)}$ such that $C = RC_\mathcal{T}$. Using Equation (A.9) we obtain

$$C \odot U = (RC_\mathcal{T}) \odot (I_d U) = (R \otimes I_d)(C_\mathcal{T} \odot U). \tag{3.55}$$

Note that the matrix $R \otimes I_d$ is invertible (since both R and I_d are invertible), hence we get the thesis. □

Remark 3.4. As explained in Appendix B, there are several types of cycle bases for a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ besides the fundamental cycle basis, namely zero-one, integral, and undirected cycle bases. In the proof of Proposition 3.6 a directed cycle basis is used since it generalizes all of them.

Equation (3.55) means that if a circuit is a linear combination of other circuits, then the compatibility constraint associated to such circuit is a linear combination of the equations associated to the addends. This implies that considering *all* the circuits in a graph is redundant and what is actually required is a maximal set of *independent* circuits (i.e. a cycle basis). In summary, we have the following result.

Corollary 3.2. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph and let C denote the cycle matrix associated to any cycle basis of $\mathcal{G}$. There exists a point formation $\mathcal{F}_\mathbf{x}$ on $\mathcal{G}$*

with directions $\mathbf{u}_{ij}$ *and lengths* α_{ij} *if and only if*

$$(C \odot U)\boldsymbol{\alpha} = \mathbf{0}. \tag{3.56}$$

Note that Equation (3.56) is the same as (3.26). Such formula captures at the same time both the structure of the graph (via C) and the specific values of the directions (via U). For example, in the case of Figure 3.5a, the matrix $C \odot U$ has the following structure

$$C \odot U = \begin{bmatrix} \mathbf{u}_{12} & \mathbf{u}_{24} & \mathbf{u}_{41} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{u}_{24} & \mathbf{0} & \mathbf{u}_{23} & \mathbf{u}_{34} \end{bmatrix} \tag{3.57}$$

where the following cycle matrix is considered

$$C = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 \end{bmatrix} \tag{3.58}$$

and the edges are ordered as in

$$U = \begin{bmatrix} \mathbf{u}_{12} & \mathbf{u}_{24} & \mathbf{u}_{41} & \mathbf{u}_{23} & \mathbf{u}_{34} \end{bmatrix}. \tag{3.59}$$

A similar formulation is derived in [179]:

$$E(C \otimes I_d)\,\mathrm{blkdiag}(\{\mathbf{u}_{ij}\}_{(i,j)\in\mathcal{E}})\boldsymbol{\alpha} = \mathbf{0} \tag{3.60}$$

where $E = \left((C \otimes I_d)(C \otimes I_d)^{\mathsf{T}}\right)^{-1/2}$. Since $(C \otimes I_d)\,\mathrm{blkdiag}(\{\mathbf{u}_{ij}\}_{(i,j)\in\mathcal{E}}) = C \odot U$ and E is invertible we get

$$\ker\left(E(C \otimes I_d)\,\mathrm{blkdiag}(\{\mathbf{u}_{ij}\}_{(i,j)\in\mathcal{E}})\right) = \ker\left(C \odot U\right) \tag{3.61}$$

which means that Equation (3.60) is equivalent to Equation (3.56). The latter enjoys a more compact formulation, which permits to exploit algebraic properties of the Khatri-Rao product, as done (e.g.) in the proof of Proposition 3.6.

Hereafter we use the notation U to denote the $d \times m$ matrix constructed from any set of directions, and we use the notation $U_{\mathbf{x}}$ to denote the $d \times m$ matrix built from the directions of a point formation $\mathcal{F}_{\mathbf{x}}$. Note that if $\mathcal{F}_{\mathbf{x}}$ is a point formation then Equation (3.56) holds, i.e. the null-space of $C \odot U_{\mathbf{x}}$ is *at least* 1-dimensional. The following theorem, which is a direct consequence of Proposition 3.4 and Corollary 3.2, states such null-space is exactly 1-dimensional if and only if $\mathcal{F}_{\mathbf{x}}$ is parallel rigid.

Theorem 3.6. *Let* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *be a connected graph and let* C *denote the cycle matrix associated to any cycle basis of* $\mathcal{G}$*. A point formation* $\mathcal{F}_{\mathbf{x}}$ *on* $\mathcal{G}$ *is parallel rigid if and only if*

$$\dim(\ker(C \odot U_{\mathbf{x}})) = 1 \tag{3.62}$$

or, equivalently, if and only if

$$\operatorname{rank}(C \odot U_{\mathbf{x}}) = m - 1. \tag{3.63}$$

3.4.1 Generic Rigidity

We now consider the property of generic rigidity. Due to Theorem 3.6 we can say that a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if $\operatorname{rank}(C \odot U_{\mathbf{x}}) = m - 1$ or, equivalently, if and only if $\dim\big(\ker(C \odot U_{\mathbf{x}})\big) = 1$, where $U_{\mathbf{x}}$ is constructed using a generic point formation $\mathcal{F}_{\mathbf{x}}$ defined on $\mathcal{G}$. Reasoning in the same way as in Section 3.3.1, we get the following algebraic characterization of generic parallel rigidity in terms of cycles in the graph.

Theorem 3.7. *A connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space if and only if*

$$\max_{\mathbf{x} \in \mathbb{R}^{nd}} \operatorname{rank}(C \odot U_{\mathbf{x}}) = m - 1, \tag{3.64}$$

or, equivalently, if and only if

$$\min_{\mathbf{x} \in \mathbb{R}^{nd}} \dim\big(\ker(C \odot U_{\mathbf{x}})\big) = 1. \tag{3.65}$$

Similarly to the node-based case [141], Theorem 3.7 can be used to develop a randomized test for checking generic parallel rigidity, where a matrix $U_{\mathbf{x}}$ is built from a point formation randomly sampled from i.i.d. Gaussian distribution. Theorem 3.7 gives also rise to an algorithm to identify maximal rigid components of flexible graphs [179], where the null-space of $C \odot U_{\mathbf{x}}$ is computed and sets of parallel lines among the rows are identified.

Note that in order to guarantee that Equation (3.64) holds, the number of rows in $C \odot U_{\mathbf{x}}$ must be greater than (or equal to) $m - 1$, i.e. the following necessary condition must be satisfied

$$d(m - n + 1) \geq m - 1 \tag{3.66}$$

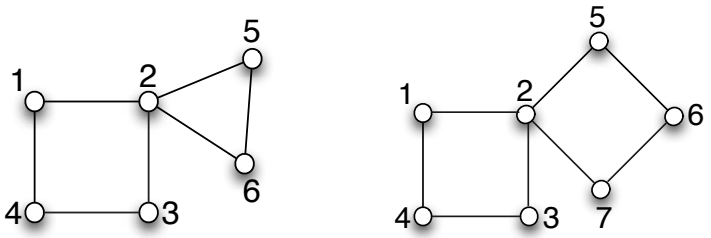
which is equivalent to Equation (3.42).

The formulation of Theorem 3.7, although equivalent to the node-based one, enables us to prove results involving the topology of the graph, showing, for instance, why triangulated graphs are rigid while graphs with long cycles may loose this property. Let us start by presenting a necessary condition for generic parallel rigidity, namely biconnectivity. This was also mentioned en-passant in [134].

Proposition 3.7. *If a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generically parallel rigid in d-space, then it is biconnected.*

Proof. If $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is not biconnected then it can be partitioned into maximal biconnected components. Let $b > 1$ denotes the number of such components and let $\mathcal{E}^1, \dots, \mathcal{E}^b$ denote their edge sets. Since the set of maximal biconnected components induces a partition of the edge set $\mathcal{E}$ and a circuit belongs to only one biconnected component, the matrix $C \odot U_{\mathbf{x}}$ can be expressed as a block-diagonal matrix (up to a re-ordering of the edges), where each block corresponds to a biconnected component, namely $C \odot U_{\mathbf{x}} = \text{blkdiag}(G_1, \dots, G_b)$. Let us assume that all such components are generically parallel rigid, otherwise the thesis is obvious, thus in each component the lengths of a generic point formation are uniquely determined by its graph and directions (up to a global scale), namely $\text{rank}(G_i) = |\mathcal{E}^i| - 1$. Thus $\text{rank}(C \odot U_{\mathbf{x}}) = \sum_{i=1}^{b}(|\mathcal{E}^i| - 1) = m - b < m - 1$, meaning that all such scales can not be reconciled into a single scale, hence $\mathcal{G}$ is generically flexible. □

It is straightforward to see that the necessary condition of Proposition 3.7 alone is not sufficient: for instance, the graph associated to the point formation in Figure 3.3a is biconnected and flexible in 2-space. However, Proposition 3.7 gives a simple condition to detect non-rigid graphs: for instance, it can be established that the graphs reported in Figure 3.6 are flexible in d-space.



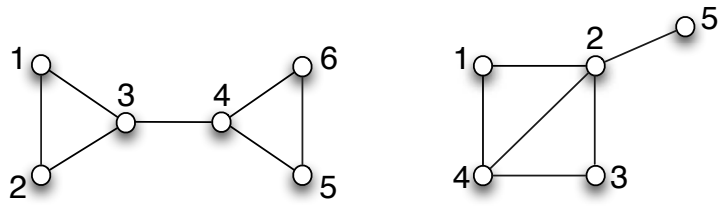
(A) (B)

FIGURE 3.6: Non biconnected graphs.

Note that if $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is biconnected (and contains at least 3 vertices), then it is bridgeless, i.e. each edge in $\mathcal{E}$ belongs to (at least) one cycle (and hence one circuit). This implies that if $\mathcal{G}$ is generically parallel rigid in d-space, then it is bridgeless. This result is not surprising since an edge not belonging to any circuit is not constrained by the other edges, and hence its length can be chosen arbitrarily. Such edge corresponds to a column of zeros in C, which makes the rank of $C \odot U_{\mathbf{x}}$ drop. Two examples of graphs with a bridge are reported in Figure 3.7.

We now consider the case where $\mathcal{G}$ consists of a single circuit of length $\ell \geq 3$, and show that short circuits are rigid while long circuits are flexible. In this case $m = n = \ell$, hence Equation (3.66) rewrites $\ell \leq d + 1$, i.e. the following proposition holds.

Proposition 3.8. *A circuit of length $\ell \geq d + 2$ is not generically parallel rigid in d-space.*



(A) (B)

FIGURE 3.7: Non bridgeless graphs.

What happens for circuits of length $\ell \leq d+1$? It can be shown that such circuits are generically parallel rigid in $\mathbb{R}^d$. Figure 3.8 reports some examples in $\mathbb{R}^3$.

Proposition 3.9. *A circuit of length $\ell \leq d+1$ is generically parallel rigid in d-space.*

Proof. If $\mathcal{G}$ consists of a single circuit then $C \odot U_{\mathbf{x}}$ is a $d \times \ell$ matrix where each column contains one direction (with the correct sign). Since a given point formation $\mathcal{F}_{\mathbf{x}}$ satisfies Equation (3.56) then

$$\operatorname{rank}(C \odot U_{\mathbf{x}}) \leq \ell - 1 \tag{3.67}$$

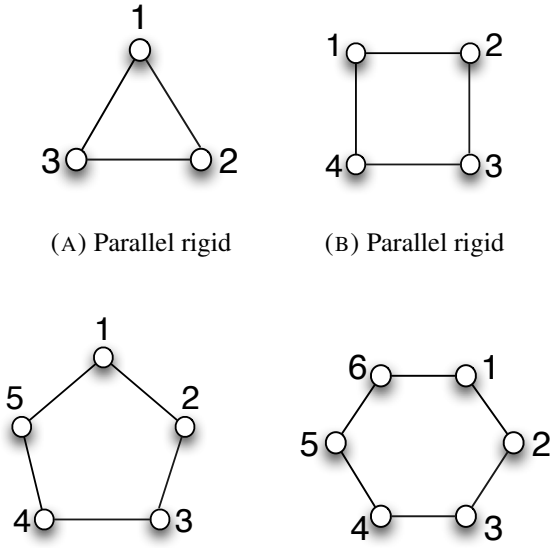
which means that the points in $\mathcal{F}_{\mathbf{x}}$ belong to an affine subspace of $\mathbb{R}^d$ of dimension at most $\ell - 1$. Note that specific configurations make $\operatorname{rank}(C \odot U_{\mathbf{x}})$ drop: $\operatorname{rank}(C \odot U_{\mathbf{x}}) = 1$ if and only if the points in $\mathcal{F}_{\mathbf{x}}$ lie on a common line; $\operatorname{rank}(C \odot U_{\mathbf{x}}) = 2$ if and only if the points in $\mathcal{F}_{\mathbf{x}}$ lie on a common plane; ... $\operatorname{rank}(C \odot U_{\mathbf{x}}) = \ell - 2$ if and only if the points in $\mathcal{F}_{\mathbf{x}}$ lie on an affine subspace of $\mathbb{R}^d$ of dimension $\ell - 2$. On the contrary, if the points in $\mathcal{F}_{\mathbf{x}}$ are generic, then $\operatorname{rank}(C \odot U_{\mathbf{x}}) = \ell - 1$, meaning that $\mathcal{F}_{\mathbf{x}}$ (and hence $\mathcal{G}$) is parallel rigid in $\mathbb{R}^d$. □

Remark 3.5. Note that there is a key difference between a circuit of length $\ell = d+1$ and a circuit of length $\ell \leq d$, which is essential to understand the next section. In the $\ell = d+1$ case, $C \odot U_{\mathbf{x}}$ is a $d \times (d+1)$ matrix, thus its rank is at most $d = \ell - 1$ *independently* of the directions, i.e. Equation (3.67) is satisfied even if $U_{\mathbf{x}}$ is substituted by a random set of $d+1$ directions. On the contrary, in the $\ell \leq d$ case, if we take a random set of ℓ directions then Equation (3.67) will not be satisfied.

Propositions 3.8 and 3.9 completely characterize the localizability of a graph made of a single circuit (in terms of its length). What happens to graphs made of several circuits? The remainder of this section reports sufficient conditions for parallel rigidity, which give some insights on how to answer such question.

Given a cycle basis for a (connected) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the *cycle graph*[2] $\mathcal{G}_C$ is defined as follow: each vertex corresponds to a circuit in the basis, and an edge is

[2]This notion generalizes the "triplet graph" of [96].



(A) Parallel rigid (B) Parallel rigid

(C) Flexible (D) Flexible

FIGURE 3.8: Rigidity of circuits in 3-space.

present between two vertices if and only if the corresponding circuits share (at least) one edge in $\mathcal{G}$. The notion of cycle graph is exploited by the following theorem.

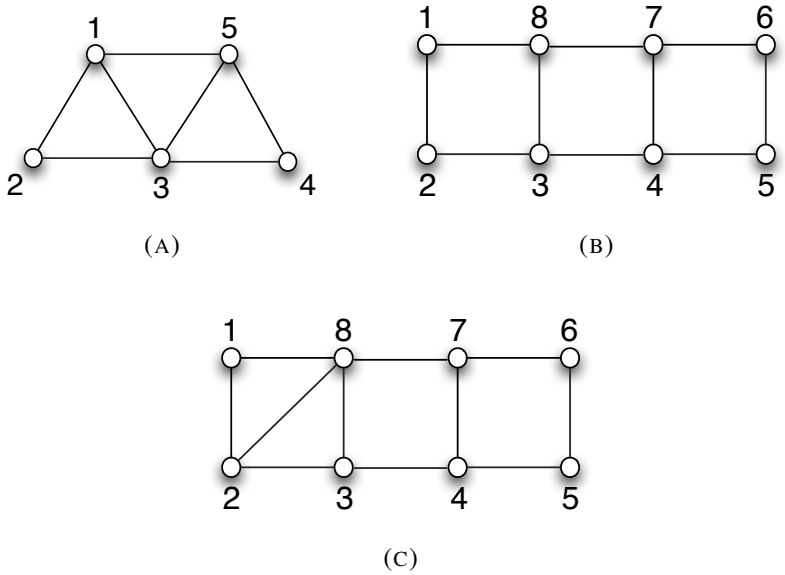
Theorem 3.8. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected bridgeless graph. Suppose that there exists a cycle basis of $\mathcal{G}$ such that the following conditions are satisfied:*

1. *each circuit in the basis has length at most $d+1$;*
2. *the associated cycle graph $\mathcal{G}_C$ is connected.*

Then $\mathcal{G}$ is generically parallel rigid in d-space.

Proof. Each circuit in the basis is generically parallel rigid in d-space due to Proposition 3.9. Since the cycle graph is connected by assumption, we can start with any circuit and reach all the others through a path, thus producing a growing rigid subgraph. Specifically, let us consider a node in $\mathcal{G}_C$ (i.e. a circuit) and let us take an edge incident to such node (i.e. a circuit sharing one edge with the first circuit). The union of these circuits is rigid since both of them are rigid and they have one edge in common. We can repeat this line of reasoning considering all the remaining circuits, obtaining a parallel rigid subgraph. Such subgraph coincides with $\mathcal{G}$ itself since $\mathcal{G}$ is bridgeless, and hence each edge in $\mathcal{E}$ belongs to (at least) one circuit in the basis. □

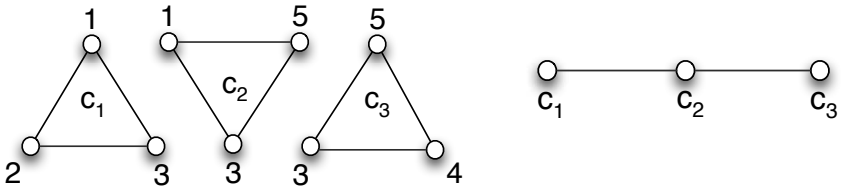
By means of Theorem 3.8 it can be established that the graphs in Figure 3.9 are parallel rigid in d-space, with $d \geq 2$ (Figure 3.9a) and $d \geq 3$ (Figures 3.9b and 3.9c). The graph in 3.9a admits a cycle basis composed of three 3-length circuits, the graph in Figure 3.9b admits a cycle basis composed of three 4-length circuits, and the graph in Figure 3.9c admits a cycle basis composed of two 3-length circuits, and two 4-length circuits. In all these cases the associated cycle graph is connected. The one associated to Figure 3.9a is reported in Figure 3.10.



(A) (B)

(C)

FIGURE 3.9: Examples of graphs which satisfy the assumptions in Theorem 3.8, and hence they are generically parallel rigid in d-space, with $d \geq 2$ (a) and $d \geq 3$ (b,c).



(A) (B)

FIGURE 3.10: Left: cycle basis composed of 3-length circuits for the graph in Figure 3.9a. Right: cycle graph $\mathcal{G}_C$ associated to such cycle basis.

Remark 3.6. Note that the reverse of Theorem 3.8 is not true. For instance, the graph reported in Figure 3.11 is parallel rigid in 3-space but Condition 1 can not be fulfilled. Let us consider the following cycle basis

$$\begin{aligned} \mathbf{c}_1 &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \\ \mathbf{c}_2 &= \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \\ \mathbf{c}_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 \end{bmatrix}^{\mathsf{T}} \end{aligned} \tag{3.68}$$

where the edges are ordered as in

$$U = \begin{bmatrix} \mathbf{u}_{12} & \mathbf{u}_{26} & \mathbf{u}_{61} & \mathbf{u}_{67} & \mathbf{u}_{73} & \mathbf{u}_{32} & \mathbf{u}_{34} & \mathbf{u}_{45} & \mathbf{u}_{51} \end{bmatrix}. \tag{3.69}$$

Note that the associated cycle graph $\mathcal{G}_C$ is connected. It is easy to see that cycle bases composed of shorter circuits do not exist, thus Condition 1 can not be satisfied. To

prove that $\mathcal{G}$ is generically parallel rigid we consider a generic point formation $\mathcal{F}_{\mathbf{x}}$ and show that its lengths are uniquely determined by its graph and directions, i.e. Equation (3.56) admits a unique solution (up to scale). Note that Equation (3.56) rewrites

$$\begin{cases} (\mathbf{c}_1 \odot U)[\alpha_{12}\ \alpha_{26}\ \alpha_{61}]^{\mathsf{T}} = \mathbf{0} \\ (\mathbf{c}_2 \odot U)[\alpha_{26}\ \alpha_{67}\ \alpha_{73}\ \ \alpha_{32}]^{\mathsf{T}} = \mathbf{0} \\ (\mathbf{c}_3 \odot U)[\alpha_{12}\ \alpha_{32}\ \alpha_{34}\ \alpha_{45}\ \alpha_{51}]^{\mathsf{T}} = \mathbf{0}. \end{cases} \tag{3.70}$$

Instead of solving Equation (3.70) globally, we follow a sequential approach. Let us start with the 3-length circuit $\mathbf{c}_1$: its lengths are uniquely determined (up to a global scale) since it is parallel rigid, meaning that, if we arbitrarily fix the value of (e.g.) α_{12}, we can (uniquely) compute the remaining lengths (i.e. α_{26} and α_{61}) by solving the first row in Equation (3.70). Then we use the obtained value of α_{26} to fix the global scale of the 4-length circuit $\mathbf{c}_2$ (which is parallel rigid), and (uniquely) solve for the remaining scales (i.e. α_{67}, α_{73} and α_{32}) by considering the second row in Equation (3.70). Note that this is possible since $\mathbf{c}_1$ and $\mathbf{c}_2$ share an edge. Finally, we consider the 5-length circuit $\mathbf{c}_3$, which is flexible (if considered in isolation). However, the key observation is that the values of α_{12} and α_{32} have been already computed, thus only three unknowns remain. In other words, the fourth row in Equation (3.70) becomes equivalent to the compatibility constraint of a 4-length circuit, and hence the remaining lengths (i.e. α_{34}, α_{45} and α_{51}) are uniquely determined. In this way we are able to compute all the unknowns up to a single scale, which corresponds to the arbitrary choice of α_{12}, meaning that $\mathcal{G}$ is parallel rigid.

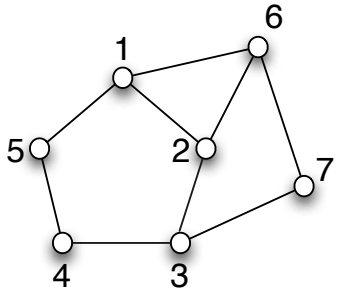


FIGURE 3.11: Example of a rigid graph in 3-space which does not satisfy the assumptions in Theorem 3.8. It admits a cycle basis composed of one 3-length circuit, one 4-length circuit and one 5-length circuit, and cycle bases with shorter circuits do not exist.

Note that the sequential approach outlined in Remark 3.6 heavily depends on the chosen cycle basis and on the order in which circuits are processed. On the contrary, if Equation (3.70) is solved globally, then any cycle basis can be used, due to Proposition 3.6.

Remark 3.6 has pointed out that a long circuit (which is flexible alone) can be part of a larger rigid graph. Indeed, if the compatibility constraint of a flexible circuit is properly combined with those of rigid circuits, it may results in a system with a unique solution (up to scale).

We conclude this section by stating the following result, which is a generalization of Theorem 3.8.

Theorem 3.9. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected bridgeless graph. Suppose that the circuits of a cycle basis can be ordered such that the first circuit has length at most $d+1$, and each of the remaining circuits satisfies one of the following conditions:*

1. *it has length at most $d+1$ and it has at least one edge in common with the subgraph induced by the previous ones;*
2. *it has length at least $d+2$, and it has at least one edge in common and at most d edges not in common with the subgraph induced by the previous ones.*

Then $\mathcal{G}$ is generically parallel rigid in d-space.

Proof. Let $\mathbf{c}_1, \ldots, \mathbf{c}_r$ denote the circuits in the basis (ordered as in the assumptions) with $r = m - n + 1$. Note that the cycle graph $\mathcal{G}_C$ is connected. The first circuit $\mathbf{c}_1$ is parallel rigid in d-space since it has length at most $d+1$. Let us consider the second circuit $\mathbf{c}_2$. If Condition 1 is satisfied, then the union of $\mathbf{c}_1$ and $\mathbf{c}_2$ is rigid, since both circuits are rigid and they have (at least) one edge in common. If Condition 2 is satisfied, then we can use the same argument as in Remark 3.6 and prove that $\mathbf{c}_2$ is equivalent to a circuit of length (at most) $d+1$ with one edge in common with $\mathbf{c}_1$, hence their union is rigid. Indeed, given a generic point formation $\mathcal{F}_\mathbf{x}$, we can first solve the compatibility constraint associated to $\mathbf{c}_1$, and arbitrarily fix the length of an edge in $\mathbf{c}_1$ in order to fix the global scale, thus all the lengths of $\mathcal{F}_\mathbf{x}$ in $\mathbf{c}_1$ are uniquely determined. Then, thanks to the edges in common with $\mathbf{c}_1$, (at most) d unknowns remain when considering the compatibility constraint associated to $\mathbf{c}_2$, which can be computed as in a circuit of length (at most) $d+1$ where one length is fixed. We can repeat this line of reasoning considering the remaining circuits $\mathbf{c}_3, \ldots, \mathbf{c}_r$ one after the other, obtaining a growing parallel rigid subgraph. Such subgraph coincides with $\mathcal{G}$ itself since $\mathcal{G}$ is bridgeless. □

It is easy to see that the graph reported in Figure 3.11 satisfies the conditions in Theorem 3.9. Establishing whether such conditions are also necessary is subject of future research.

3.5 The Parallel Rigidity Index

The results on generic rigidity reported in Sections 3.3 and 3.4 are derived considering a generic point formation $\mathcal{F}_\mathbf{x}$, which uniquely defines a set of directions (i.e. a measurement function). What happens if the *directions*, not the coordinates of the points in $\mathcal{F}_\mathbf{x}$, are generic? This issue is addressed in [146], where the authors, starting from the edge-based formulation, introduce the concept of parallel rigidity index, which is a property of the graph and dimension d.

Definition 3.7. The *parallel rigidity index* $\mathcal{I}_d(\mathcal{G})$ of a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in d-space is defined as

$$\mathcal{I}_d(\mathcal{G}) = \min_{U \in \mathbb{R}^{d \times m}} \dim \big(\ker(C \odot U) \big). \tag{3.71}$$

Note that there is an essential difference between the right side in Equation (3.71) and the left side in Equation (3.65): the former computes the minimum over all the possible edge directions (contained in U) whereas the latter computes the minimum over all possible point formations $\mathcal{F}_\mathbf{x}$ (which define a matrix $U_\mathbf{x}$).

Remark 3.7. An equivalent definition for the parallel rigidity index is the following

$$\mathcal{I}_d(\mathcal{G}) = m - \max_{U \in \mathbb{R}^{d \times m}} \operatorname{rank}(C \odot U). \tag{3.72}$$

We observe that the above equation does not coincide, in general, with $m - \operatorname{ger}(C \odot U)$, where $\operatorname{ger}(C \odot U)$ denotes the *generic rank* [114] of $C \odot U$, that is the maximal rank that $C \odot U$ (viewed as a structured matrix) achieves as a function of its arbitrary (non-zero) elements. Indeed, when considering Equation (3.72), the nonzero entries in $(C \odot U)$ are not arbitrary at all: for instance, if an edge (i, j) belongs to more than one circuit in C, then multiple copies of $\mathbf{u}_{ij}$ appear in $C \odot U$, one for each circuit. It is easy to see that

$$\mathcal{I}_d(\mathcal{G}) \geq m - \operatorname{ger}(C \odot U). \tag{3.73}$$

The generic rank has also a combinatorial description [114], namely it is equal to the maximum number of edges of any matching of a bipartite graph constructed as follows: nodes correspond to rows/columns of the matrix, and edges correspond to its nonzero entries. Establishing under which conditions (if any) equality holds in (3.73) is left to future research.

It can be shown that the minimum in Equation (3.71) is attained for generic directions [146], where a set of directions is called generic if its dn coordinates are not algebraically dependent. Thus we can rewrite the parallel rigidity index as $\mathcal{I}_d(\mathcal{G}) = \dim \big(\ker(C \odot U) \big)$, or, equivalently,

$$\mathcal{I}_d(\mathcal{G}) = m - \operatorname{rank}(C \odot U) \tag{3.74}$$

where U is a $d \times m$ matrix containing *generic* directions in its columns. This suggests a randomized procedure to compute the parallel rigidity index, where a matrix U is built from a set of directions sampled at random on the sphere in d-space, similarly to the randomized test for parallel rigidity proposed in [141].

As observed in Section 3.4, a point formation $\mathcal{F}_\mathbf{x}$ satisfies Equation (3.56), i.e. the null-space of $C \odot U_\mathbf{x}$ is at least 1-dimensional. On the contrary, if we consider generic directions we can not expect to find a non-trivial solution to $(C \odot U)\boldsymbol{\alpha} = \mathbf{0}$ for any graph $\mathcal{G}$, i.e. it may happen that a point formation on $\mathcal{G}$ with such directions does not exist. In other words, the parallel rigidity index $\mathcal{I}_d(\mathcal{G})$ can be equal to zero, which means that the only length assignment compatible with a generic set of m

directions is $\boldsymbol{\alpha} = \mathbf{0}$, i.e. all the nodes collapse into one point in $\mathbb{R}^d$. $\mathcal{I}_d(\mathcal{G}) = 1$ means that there exists a unique (up to scale) length assignment $\boldsymbol{\alpha} \neq \mathbf{0}$ compatible with a generic set of directions, i.e. for *any* set of directions in $\mathbb{R}^d$ (with coordinates not algebraically dependent) there exists a unique (up to translation and scale) point formation $\mathcal{F}_\mathbf{x}$ on $\mathcal{G}$ (such that not all the nodes are coincident) having such directions as measurement function. $\mathcal{I}_d(\mathcal{G}) \geq 2$ means that there exists a point formation $\mathcal{F}_\mathbf{x}$ on $\mathcal{G}$ compatible with a generic set of directions, but it is not unique (up to translation and scale), i.e. there are additional degrees of freedom.

Using the definition, we can easily compute the parallel rigidity index of a graph $\mathcal{G}$ consisting of a single circuit of length $\ell \geq 3$. The difference between the $\ell \leq d$ case and the $\ell = d+1$ case has been already observed in Remark 3.5.

Proposition 3.10. *Let $\mathcal{G}$ be a circuit of length ℓ.*

- *If $\ell \leq d$ then $\mathcal{I}_d(\mathcal{G}) = 0$;*
- *if $\ell = d+1$ then $\mathcal{I}_d(\mathcal{G}) = 1$;*
- *if $\ell \geq d+2$ then $\mathcal{I}_d(\mathcal{G}) = \ell - d \geq 2$.*

Proof. If $\mathcal{G}$ is a circuit of length ℓ, then $m = \ell$ and $C \odot U$ is a $d \times \ell$ matrix containing any (generic) direction in each column. Thus $C \odot U$ has full rank and, using (3.74), we get $\mathcal{I}_d(\mathcal{G}) = \ell - \min\{d, \ell\}$. □

Using Proposition 3.10, we get that $\mathcal{I}_3(\mathcal{G}) = 0$ for the graph in Figure 3.8a, $\mathcal{I}_3(\mathcal{G}) = 1$ for the graph in Figure 3.8b, $\mathcal{I}_3(\mathcal{G}) = 2$ for the graph in Figure 3.8c, and $\mathcal{I}_3(\mathcal{G}) = 3$ for the graph in Figure 3.8d.

The following result provides a combinatorial characterization of the parallel rigidity index of a connected graph.

Theorem 3.10 ([146])**.** *The parallel rigidity index $\mathcal{I}_d(\mathcal{G})$ of a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in d-space is equal to the minimal size of the intersection of d spanning trees of $\mathcal{G}$.*

Corollary 3.3. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph, let n_b denotes the number of bridges, and let $\mathcal{G}^1, \ldots, \mathcal{G}^{n_b}$ denote the connected components obtained after removing all the bridges from $\mathcal{G}$. Then*

$$\mathcal{I}_d(\mathcal{G}) = n_b + \sum_{i=1}^{n_b} \mathcal{I}_d(\mathcal{G}^i). \tag{3.75}$$

Proof. Note that, if $\mathcal{G}$ has a bridge, then this bridge belongs to any spanning tree of $\mathcal{G}$. Recall that the removal of a bridge disconnects the graph, and hence it can be partitioned into connected components. Thus any minimal intersection of d spanning trees of $\mathcal{G}$ is the union of all the bridges and a minimal intersection of d spanning trees for each connected component. □

According to Theorem 3.10, the parallel rigidity index can be computed by first counting the edges in common between d (distinct) spanning trees, and then taking the minimum over all the possible choices of such spanning trees. In this way it can be established, for instance, that $\mathcal{I}_d(\mathcal{G}) = 0$ for the graphs in Figures 3.5a and 3.9a. Using Corollary 3.3 we get that $\mathcal{I}_d(\mathcal{G}) = 1$ for the graphs in Figure 3.7. In general, $\mathcal{I}_d(\mathcal{G})$ is greater than or equal to the number of bridges of $\mathcal{G}$. Note that adding an edge between existing nodes may modify the parallel rigidity index. For instance, as it can be easily verified, the graph in Figure 3.9b satisfies $\mathcal{I}_3(\mathcal{G}) = 1$ whereas the graph in Figure 3.9c (which is obtained from the former by adding one edge) satisfies $\mathcal{I}_3(\mathcal{G}) = 0$.

The following result, which is a straightforward consequence of Theorem 3.10, establishes the relation between the parallel rigidity index in d-space and in $(d+1)$-space.

Corollary 3.4 ([146]). *The parallel rigidity index $\mathcal{I}_d(\mathcal{G})$ of a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in d-space decreases as d grows, namely $\mathcal{I}_d(\mathcal{G}) \geq \mathcal{I}_{d+1}(\mathcal{G})$.*

What is the relation between the parallel rigidity index and the generic rigidity of a graph? Since the set of directions coming from point formations is contained in the set of all possible directions, we get

$$\min_{U \in \mathbb{R}^{d \times m}} \dim \big(\ker(C \odot U) \big) \leq \min_{\mathbf{x} \in \mathbb{R}^{nd}} \dim \big(\ker(C \odot U_{\mathbf{x}}) \big). \tag{3.76}$$

Thus, if $\mathcal{I}_d(\mathcal{G}) \geq 2$ then $\mathcal{G}$ is generically flexible in d-space. In other words, if $\mathcal{G}$ is generically parallel rigid in d-space, then either $\mathcal{I}_d(\mathcal{G}) = 0$ or $\mathcal{I}_d(\mathcal{G}) = 1$. The converse is not true, i.e. the parallel rigidity index of a flexible graph can assume any value. For example, the graphs reported in Figure 3.6 are both flexible in 3-space and – as it can be easily verified – the parallel rigidity index is $\mathcal{I}_d(\mathcal{G}) = 1$ for the left sub-figure and $\mathcal{I}_d(\mathcal{G}) = 2$ for the right sub-figure. The flexible graph in Figure 3.5b satisfies $\mathcal{I}_d(\mathcal{G}) = 0$. More examples are reported in Figure 3.12.

3.5.1 Which rigidity for Error Compensation?

Let us now come back to the network localization problem. As explained in Section 3.2.2, given a set of directions $\{\mathbf{u}_{ij}\}$ and a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the unknown node locations $\mathbf{x}_j \in \mathbb{R}^d$ can be recovered via a two-step procedure: first, the unknown lengths α_{ij} are computed by solving system (3.56); then, the unknown node locations are derived as the solution of Equation (3.3). Other methods can also be found in the literature, which refer to the node-based formulation (e.g. [78, 33]).

Let us consider the noiseless case where $\mathbf{u}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/||\mathbf{x}_i - \mathbf{x}_j||$. In this scenario the graph is required to be generically parallel rigid in d-space, in order to guarantee that Equation (3.56) has a unique solution (up to scale), and hence the network localization problem, i.e. Equation (3.3), admits a unique solution (up to translation and scale).

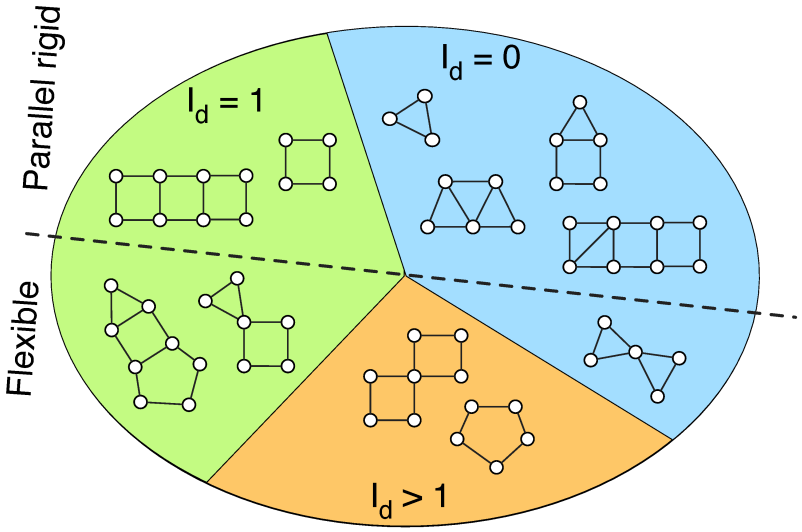


FIGURE 3.12: Relationship between the notion of generic parallel rigidity and the parallel rigidity index $\mathcal{I}_d(\mathcal{G})$ with $d = 3$. The graphs amenable to error compensation are the ones in the upper right corner.

Suppose now that measurements are noisy (as it is the always the case in practice), i.e. $\mathbf{u}_{ij} \approx (\mathbf{x}_i - \mathbf{x}_j)/||\mathbf{x}_i - \mathbf{x}_j||$. In this case, besides parallel rigidity, one has a further requirement, which is related to error compensation. In particular, one wants to exclude graphs that, by virtue of their structure, always yield an exact solution to the network localization problem *regardless* of the direction measures. This analysis cannot be performed within the generic rigidity framework, for it is based on generic point formations (which always yield exact directions), but the parallel rigidity index is required, which relies on generic directions.

As a matter of fact, the graphs which, due to their structure, satisfy $\dim\big(\ker(C \odot U)\big) = 1$ for *any* (generic) U are all those and only those with $\mathcal{I}_d(\mathcal{G}) = 1$, thus there exists an exact solution to Equation (3.56) in the presence of noise and/or outliers. On the contrary, $\mathcal{I}_d(\mathcal{G}) = 0$ means that $\dim\big(\ker(C \odot U)\big) = 0$ and no exact solution to Equation (3.56) exists (apart from the trivial solution $\boldsymbol{\alpha} = \mathbf{0}$), as one expects in a noisy case.

We conclude that: *among all the graphs which are generically parallel rigid in d-space, only the ones with* $\mathcal{I}_d(\mathcal{G}) = 0$ *are amenable to error compensation.* They correspond to the the upper right corner of Figure 3.12.

3.6 Conclusion

In this chapter we considered the localizability problem of a sensor network in d-space constrained with direction measures, which is studied under the name of parallel rigidity. We provided a unifying view of such a problem: first, we reviewed the node-based formulation of parallel rigidity; then, we described the edge-based formulation, which is equivalent to the node-based one; finally, we suggested how the parallel rigidity index permits to identify which graphs promote error compensation

in bearing-based network localization. Section 4.4 will explain how this theory can be profitably applied to structure from motion.

As concerns possible future work, several directions could be investigated. First, we aim at establishing if the sufficient conditions in Theorem 3.9 are also necessary, and, in case of a negative answer, we aim at finding a characterization of localizability in terms of a cycle basis (based on the length of its circuits and how they overlap). Secondly, we will explore under which assumptions (if any) the parallel rigidity index can be rewritten in terms of the generic rank of $C \odot U$. Finally, from the practical perspective, we plan to compare the node-based and edge-based approaches in the presence of noise. In this context, we also aim at studying whether the choice of a particular cycle basis influences the performances of the edge-based localization.

Chapter 4

Applications

In this chapter we show how the framework of synchronization can be profitably used in several Computer Vision problems. Among the applications that have been mentioned in Chapter 2 (image mosaicking, multi-view matching, clock synchronization, etc.) we concentrate here on a few ones where the formulation of the problem in terms of synchronization might require some clarification. Particular attention is given to the structure from motion problem, which is tightly related also to bearing-based localization.

4.1 Introduction

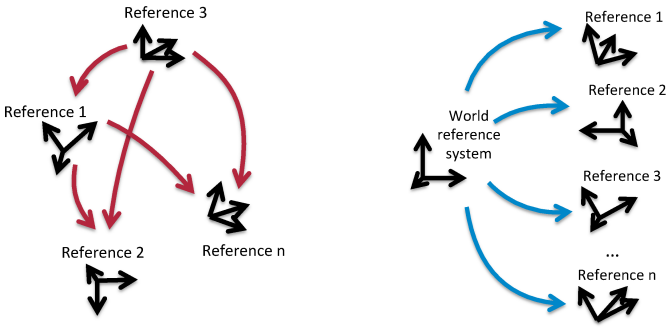
The synchronization problem, defined in Chapter 2, requires to find elements of a group (or, more generally, of an inverse monoid) given a redundant set of measures of their ratios (or differences), which are represented as a measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In this chapter we are particularly interested in the Symmetric Group $Sym(d)$ (and Inverse Semigroup $ISym(d)$) and in the Special Euclidean Group $SE(d)$.

Elements of $Sym(d)$ represent matches between feature sets, that are typically extracted from a set of images. Synchronization of these matches is tantamount to joining them in multi-view correspondences while enforcing loop-closure constraints. This is called *joint matching* or *multi-view matching* by some authors. In practical scenarios not all the features are visible (or matchable) in all the images, so each matching is modelled as an element of $ISym(d)$.

Elements of $SE(d)$ represent the angular attitude and position of a d-dimensional reference frame. These two properties are collectively referred to as *motion* in Computer Vision, *orientation* in Photogrammetry, or *pose* in Robotics. Synchronization over $SE(d)$ is tantamount to recovering the location and attitude of a set of reference frames organized in a network, where the links of this network are relative transformations of one frame with respect to (some of) the others, as shown in Figure 4.1. This is also called *network orientation* [69], or *sensor network localization* [55], or *motion averaging* [79], or *pose-graph optimization* [41]. If we restrict the attention to the angular attitude (leaving out the position) then we get a rotation synchronization. Similarly, if position only is considered, it results in translation synchronization.

Such local frames can be local coordinates where $3D$ points are represented, in which case we are dealing with *multiple point-set registration* [147], or camera

reference frames, in which case we are in the context of *structure-from-motion* [142]. In the first case, the goal is to find the rigid transformations that bring multiple 3D point sets into alignment, whereas in the second case the goal is to recover both scene structure (3D coordinates of scene points) and camera motion starting from a set of images. Other applications include structural biology [168], where the problem of recovering the three-dimensional structure of a macromolecule from many cryo-electron microscopy (cryo-EM) images is considered, and simultaneous localization and mapping (SLAM) [30, 151, 41], where the goal is to localize a robot moving in an unknown environment while building a map of the environment.



(A) Input (B) Output

FIGURE 4.1: Synchronization of frames.

Note that the motion task of the structure-from-motion problem cannot be straightforwardly solved as a rigid-motion synchronization, due to the depth-speed ambiguity. Indeed, only the *directions* of the relative displacements between camera pairs can be measured, but the magnitude is unknown. In other words, camera positions are computed from pairwise directions, and this is an instance of bearing-based network localization in 3-space, where sensors are the cameras, which can be solved via a node-based approach (as explained in Section 3.2.1) or an edge-based approach (as done in Section 3.2.2), where the unknown magnitudes (also called *epipolar scales*) are recovered.

4.1.1 Contribution

In this chapter we show how some Computer Vision problems can be addressed in terms of synchronization. In particular, Section 4.2 is devoted to multi-view matching, Section 4.3 analyzes multiple point-set registration, and Section 4.4 considers structure from motion. Particular attention is given to the latter, where the formulation of the problem in terms of synchronization is not trivial due to the depth-speed ambiguity.

Specifically, in Section 4.4.2 we point out the connection between structure from motion and bearing-based localization and in Section 4.4.4 we show how the approach detailed in Section 3.2.2 can be generalized to the case where a global coordinate system is not available, thus allowing to recover the epipolar scales based on the knowledge of relative rotations and translation directions.

We also present a scalable pipeline which estimates camera motion starting from the relative motions of a subset of camera pairs, which is grounded on the concept of synchronization (Section 4.4.3). The pipeline is composed of three stages. We first compute absolute rotations by solving a synchronization problem over $SO(3)$, using the spectral solution described in Section 2.7. In the second stage, we estimate the epipolar scales by partitioning the graph into smaller subgraphs: we compute the translation magnitudes locally, as done in Section 3.2.2, and then we globally derive all the scales by solving a synchronization problem over $\mathbb{R}$. In the third stage, we recover absolute translations by solving a synchronization problem over $\mathbb{R}^3$. All the considered synchronization instances translate into direct solutions, namely eigenvalue decompositions (rotations and scales) or linear least squares (translations), which are coupled with IRLS to gain robustness to outliers.

4.2 Multi-view Matching

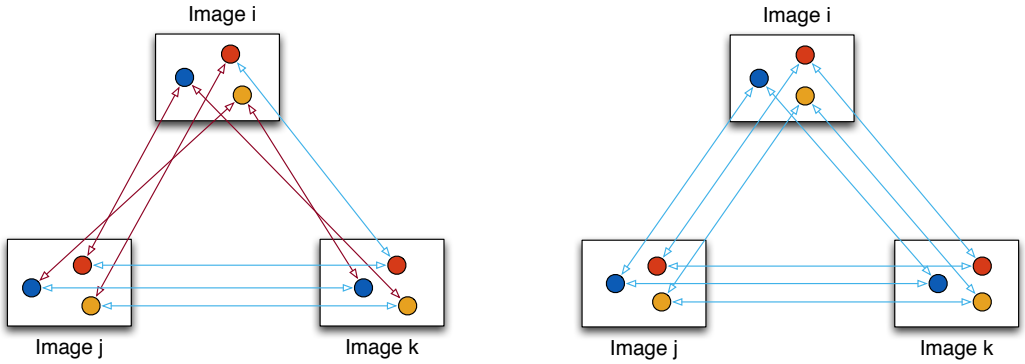
Establishing correspondences between feature sets is a fundamental problem in computer vision, that lies at the basis of any geometric computation (e.g., structure from motion) and also object recognition and shape analysis. We are particularly interested in the case where features are extracted from a collection of images.

The majority of the works on this topic focus on finding correspondences between two feature sets [120, 112, 117, 115, 121, 149]. However, in many tasks it is often required to find matches across multiple views. Moreover, recent studies have suggested that jointly optimizing the correspondences across the whole dataset can lead to significant improvements when compared to computing matches between pairs of views in isolation [143, 205], since pairwise matching algorithms can generate noisy and unreliable results.

As a matter of fact, appearance and geometry alone cannot guarantee the correctness of the matches, hence all one can do is to resort to higher level constraints that arise from the closed-loop consistency of matching across multiple views. This is called *joint matching* or *multi-view matching* by some authors.

A natural approach to joint matching consists in operating directly in the feature space, namely optimizing a cost function which explicitly depends on the features extracted in all the images. Early solutions of this type include the methods presented in [155, 157, 123, 161]. More recently, the authors of [89] cast multi-view matching to an image indexing problem, while in [51] a game-theoretical approach is adopted and the matching problem is expressed as a non-cooperative game. Rank constraints are introduced in [137] for point matching across video frames, and this approach is

extended in [200, 95], where the joint matching problem is robustly formulated as a low-rank and sparse matrix decomposition.



(A) Input (B) Output

FIGURE 4.2: Multi-view matching.

A different approach is adopted in [92, 47, 205, 127, 143, 164, 197] where multi-view matching is solved in two steps, as shown in Figure 4.2: first, matching between pairs of images is performed in isolation; then, such correspondences are improved by globally optimizing their internal coherence, without relying on the actual value of the features. These methods are usually faster and less memory-demanding than feature-based ones.

In [92] a solution based on semidefinite programming is proposed, which, however, assume total feature correspondences between all images. Such technique is extended in [47] in order to handle partial correspondences, and theoretical guarantees for exact matching in the presence of corrupted input are provided, assuming a certain noise model. In [205] the joint matching problem is formulated as a low-rank matrix recovery task and the nuclear-norm relaxation for rank minimization is employed, whereas in [127] a practical and efficient method is proposed based on spectral decomposition and an approximate strategy for projection onto $ISym(d)$. The authors of [143, 164, 197] express multi-view matching as a synchronization problem, which is solved via spectral decomposition [143, 164] or the Gauss-Seidel method [197]. However, as [92], total correspondences between all the images are assumed, thus limiting their applicability to real scenarios.

Problem Formulation

Consider a set of n *nodes*. A set of k_i *objects* out of d is attached to node i (we say that the node "sees" these k_i objects) in a random order, i.e., each node has its own local labeling of the objects with integers in the range $\{1, \ldots, d\}$.

For example, with reference to Fig. 4.3, the same object is referred to as n. 5, e.g., in node A and as n. 3 in node B. It is assumed that pairs of nodes can *match* these objects, establishing which objects are the same in the two nodes, despite the different naming. For example, a match means that the two nodes agree that "my

object n. 5 is your object n. 3". The goal is to infer a global labeling of the objects, such that the same object receives the same label in all the nodes.

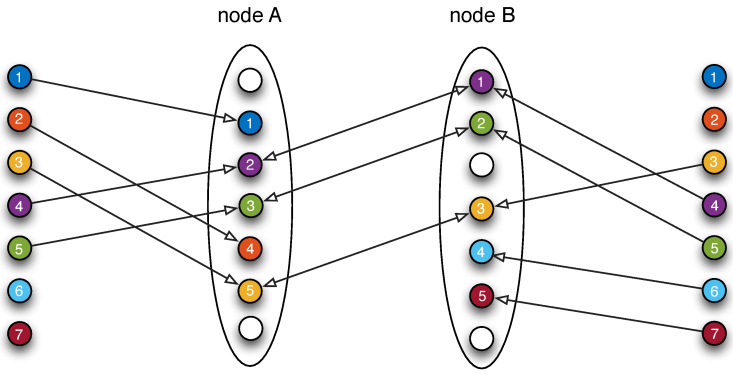


FIGURE 4.3: In the center, two nodes with partial visibility match their three common objects. At the extrema the ground truth ordering of the objects. Each node sees some of the objects (white circles are missing objects) and puts them in a different order, i.e., it gives them different numeric labels.

A more concrete problem statement can be given in terms of feature matching, where nodes are *images* and objects are *features*. A set of matches between pairs of images is computed in isolation (via e.g. SIFT [120]), and the goal is to jointly update them so as to maximize their consistency. In general, not all the features are visible (or matchable) in all the images, so each matching is modelled as a partial permutation, that is an element of $ISym(d)$. Otherwise, total permutations are used, that are elements of $Sym(d)$.

Specifically, the partial permutation matrix P representing the matching between node B and node A is constructed as follows: $[P]_{h,k} = 1$ if object k in node B is matched with object h in node A; $[P]_{h,k} = 0$ otherwise. If row $[P]_{h,\cdot}$ is a row of zeros, then object h in node A does not have a matching object in node B. If column $[P]_{\cdot,k}$ is a column of zeros, then object k in node B does not have a matching object in node A. Figure 4.4 shows one example.

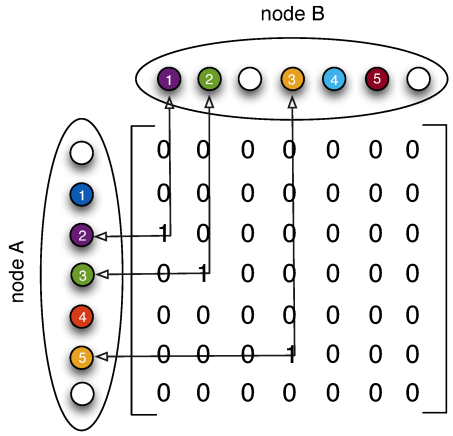


FIGURE 4.4: Partial permutation matrix representing the matching between two nodes.

Let $P_{ij} \in ISym(d)$ denote the partial permutation representing the matching between node j and node i, and let $P_i \in ISym(d)$ (resp. $P_j \in ISym(d)$) denote the unknown partial permutation that reveals the true identity of the objects in node i (resp. j). The matrix P_{ij} is called the *relative* permutation of the pair (i, j), and the matrix P_i (resp. P_j) is called the *absolute* permutation of node i (resp. j). It can be easily verified that

$$P_{ij} = P_i P_j^{\mathsf{T}}. \tag{4.1}$$

Thus the problem of finding the global labeling can be modeled as finding n absolute permutations assuming that a set of relative permutations is known, where the link between relative and absolute permutations is given by Equation (4.1).

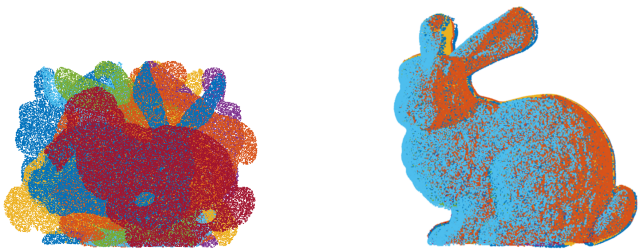
If permutations were total, Equation (4.1) would be recognized as the consistency constraint of a synchronization problem over $Sym(d)$ [143, 164, 197], which can be solved in closed form either via spectral decomposition or singular value decomposition, as explained in Section 2.9. However, in all practical settings, permutations are partial, thus the synchronization problem over the inverse monoid $ISym(d)$ has to be addressed. This can be done via the spectral solution that we conceived in Section 2.10 as an extension of [143, 164] to the case of partial permutations.

Note that the methods in [92, 47, 205, 127] do not solve a synchronization problem, since they compute relative permutations instead of absolute ones. While in a group it is always possible to recover vertex labels from edge labels, by setting one node equal to the identity and propagating Equation (2.6) along a spanning tree, this procedure does not apply to an inverse monoid, since such relation is no longer equivalent to the consistency constraint of synchronization, namely Equation (2.5). For this reason, such techniques are not analyzed in this thesis.

4.3 Multiple Point-set Registration

The goal of multiple point-set registration is to find the rigid transformations that bring multiple ($n \geq 2$) 3D point sets into alignment, as shown in Figure 4.5, where each rigid transformation is represented by a direct isometry, i.e. an element of $SE(3)$. Such point sets usually come from a 3D scanning device, which can frame only a fraction of an object from a given viewpoint. Therefore, registration of multiple scans is necessary to build a full 3D model of the object. This problem covers a wide range of applications, including (but not limited to) cultural heritage, engineering modelling and virtual reality.

If $n = 2$ then we are dealing with a *pairwise* (two point-sets) registration problem. The gold standard in this context is the Iterative Closest Point (ICP) Algorithm [22, 48], which computes correspondences between the point sets given an estimate for the rigid transformation, then updates the transformation based on the current correspondences, and iterates through these steps until convergence – to a local minimum – is reached. See [153] for several variants of the ICP Algorithm.



(A) Input (B) Output

FIGURE 4.5: Multiple point-set registration.

If $n > 2$ then we are dealing with a *multiple* point-set registration problem, which is more complex than the $n = 2$ case due to the high amount of parameters that have to be estimated. Among the initial attempts to address this problem are the *sequential* techniques introduced in [48, 147], that repeatedly register a new point set into a growing model, until all the sets are considered. This approach however does not take into account all the available constraints, e.g. the constraint between the last and first point set is not used if the sets are obtained using a turntable, therefore the solution is suboptimal. A different paradigm is realized by *global* methods, which are able to register simultaneously all the points sets. They are able to exploit the redundancy in the constraints between pairs of point sets, to compensate and distribute the error, thereby preventing drift in the solution.

Global registration can be solved in *point space* or in *frame space*. In the first case, all the rigid transformations are computed by optimizing a cost function that depends on the distance between corresponding points. In the second case, the optimization criterion is related to the internal consistency of the network of rigid transformations applied to the local coordinate frames.

Early point-space solutions include the methods presented in [193, 144, 18, 19]. More recently, [109] solves the problem on the manifold of rotations with Gauss-Newton iterations and then computes translations through least-squares. Such approach is improved in [27] by reducing computational time and it is embedded in a Bayesian framework in [128], in order to take into account reliability of correspondences. A similar formulation is adopted in [46] where the authors cast the registration problem to a semidefinite program, proving conditions for exact and stable recovery of rigid transformations. In [148] a robust solution is derived by minimizing a cost function based on the ℓ_1-norm. A generalization of the ICP Algorithm to $n > 2$ is described in [65], which builds on the Levenberg-Marquardt ICP formulation of [67], while in [176] multiple point-set registration is solved by combining ICP and Generalized Procrustes Analysis [16]. A related approach employs rank minimization for global registration of multiple depth images [175].

Frame-space methods originate from the pioneering works of [163] and [71]. More recent solutions comprise [177, 82, 20, 10, 9]. Frame-space methods are faster and less memory-demanding than point-space ones. Unquestionably, any optimal

formulation *must* include points in the cost function, in analogy to bundle adjustment in the context of structure-from-motion. Nevertheless, frame-space approaches yield a fairly accurate registration.

At the border between frame-space and point-space methods is the formulation in [166], where 3D points are used to compute a second-order approximation of the cost function, but they are not involved in subsequent computations.

Among the aforementioned methods, [109, 27, 128, 46, 163] first recover the rotation component of the rigid transformations and then compute translations, whereas [148, 65, 176, 175, 71, 177, 79, 20, 166] compute rotations and translations simultaneously, as elements of SE(3).

Problem formulation

Let $\mathcal{P} = \{\mathbf{p}_k\}_{k=1}^{v}$ be a set of 3D points representing a given object expressed in an absolute (world) coordinate system. Let $\{\mathcal{P}^i\}_{i=1}^{n}$ denote multiple views of the object taken from different positions and viewing directions, where each 3D point set $\mathcal{P}^i = \{\mathbf{p}_k^i\}_{k \in V_i}$ refers to a subset $V_i \subseteq \{1, \ldots, v\}$ of the original v points. Let $M_i \in SE(3)$ denote the 3D displacement between the local reference frame of view i and the world coordinate system, which is referred to as the *absolute motion* of view i, namely

$$M_i = \begin{pmatrix} R_i & \mathbf{t}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \in SE(3) \tag{4.2}$$

where $R_i \in SO(3)$ represents the rotation component of the transformation, and $\mathbf{t}_i \in \mathbb{R}^3$ represents the translation component. Using this notation, the (homogeneous) coordinates of the k-th point can be expressed in the reference frame of view i as

$$\mathbf{p}_k^i = M_i \mathbf{p}_k \tag{4.3}$$

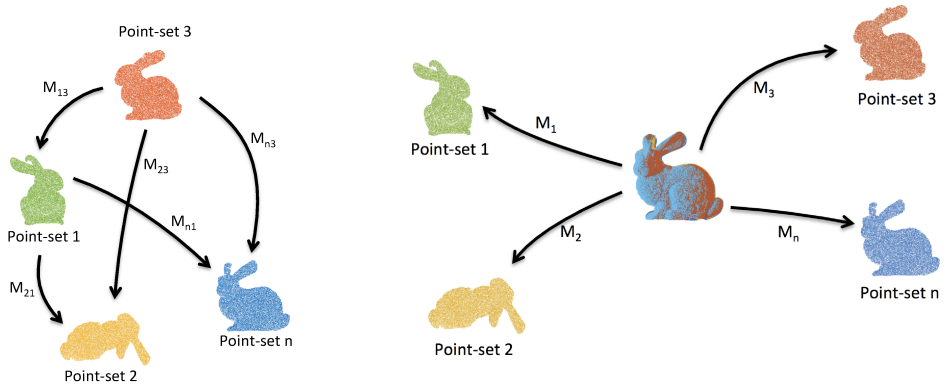
and the relation between the coordinates of $\mathbf{p}_k$ in references i and j is given by

$$\mathbf{p}_k^i = M_i M_j^{-1} \mathbf{p}_k^j \tag{4.4}$$

assuming that $k \in V_i \cap V_j$, where the index set $V_i \cap V_j$ defines corresponding points between $\mathcal{P}^i$ and $\mathcal{P}^j$.

The goal of multiple point-set registration is to estimate the absolute transformations $M_i \in SE(3)$ starting from the knowledge of the point sets $\{\mathcal{P}^i\}_{i=1}^{n}$. Since $\mathcal{P}$ can be recovered from Equation (4.3) by applying the inverse of absolute motions to each point, the absolute motions can be viewed as the transformations that bring multiple point sets into alignment. The index sets $\{V_i\}_{i=1}^{n}$ are in general unknown, and therefore they have to be computed beforehand or during the registration process.

The registration problem can be profitably formulated in frame space without involving 3D points [163, 71, 177, 82, 20], as shown in Figure 4.6. Let $M_{ij} \in SE(3)$ denote the rigid transformation between the reference frame of view i and that of view j, which is referred to as the *relative motion* of the pair (i, j). It follows from



(A) Input (B) Output

FIGURE 4.6: Frame-space multiple point-set registration.

Equation (4.4) that the following condition holds

$$M_{ij} = M_i M_j^{-1} \tag{4.5}$$

which means that the registration problem can be reduced to finding the absolute motions starting from a subset of relative motions. This is exactly an instance of *rigid-motion synchronization* where the input edge labels are typically computed via the ICP algorithm. Note that outliers may appear, caused by failure of ICP to estimate the correct transformation between two point-sets, which in turn may be originated by insufficient overlap and/or poor initialization. Thus multiple-point set registration can be solved via the spectral or null-space solutions, introduced in Section 2.8, enhanced with IRLS to gain robustness to outliers. Alternatively, the LRS formulation can be used, that naturally includes missing data and outliers in its definition, as explained in Section 2.5.5.

4.4 Structure from Motion

Recovering geometric information about a scene captured by multiple cameras has a great relevance in Computer Vision. In the structure from motion problem such geometric information includes both scene structure, i.e., 3D coordinates of scene points, and camera motion, i.e., absolute positions and attitudes of the cameras, as shown in Figure 4.7. This problem also appears in the context of Photogrammetry under the name of *network orientation*. Several systems have been developed to reconstruct large-scale scenes from a collection of unordered images, recently surveyed in [142]. They can be divided into three categories: *structure-first*, *structure-and-motion*, *motion-first*.

Structure-first approaches (e.g., [53]) begin with estimating the structure and then compute the motion. Specifically, stereo-models are built and co-registered, similarly

(A) Input (B) Output

FIGURE 4.7: Structure from Motion.

to the multiple point-set registration problem.

Structure-*and*-motion techniques solve simultaneously for structure and motion. Bundle block adjustment [178, 72], resection-intersection (sequential) methods [169, 34, 196, 158], and hierarchical methods [74, 135] belong to this category. Although being highly accurate, these approaches suffer from two main disadvantages: on one hand they require intermediate expensive non-linear minimizations to damp error propagation, on the other hand the output may depend on the order in which images are added or on the choice of the initial pair/triplet.

Motion-first methods initially recover the motion and then compute the structure [78, 126, 52, 4, 134, 96, 195, 181, 140, 76, 6]. In this thesis we are specifically interested in frame-space methods, that do not make use of points in order to compute the motion. They start from the relative motions determined from point matches among the images, they compute the angular attitude and position of the cameras with respect to an absolute coordinate frame, and they return a sparse 3D point cloud representing the scene. These motion-first methods are *global*, for they take into account the entire relative information at once, or, in other terms, they consider the whole *epipolar graph* [134, 5] (also known as the *viewing graph* [113]), where the nodes correspond to the cameras and the edges represent epipolar (i.e. two-view) relationships. Thus motion-first methods have the advantage of fairly distributing the errors among the cameras, and thus they need bundle adjustment only at the end, thereby resulting in a reduction of the computational cost. For this reason they have gained increasing attention in the community, similarly to frame-space methods for multiple point-set registration.

4.4.1 Problem Formulation

Consider n pinhole cameras that capture the same (stationary) 3D scene. Let $M_i \in SE(3)$ denote the rigid transformation between the local reference frame of camera i and the world coordinate system, which is referred to as the *absolute motion* of camera i, namely

$$M_i = \begin{pmatrix} R_i & \mathbf{t}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \in SE(3) \tag{4.6}$$

where $R_i \in SO(3)$ represents the rotation component of the transformation, and $\mathbf{t}_i \in \mathbb{R}^3$ represents the translation component. The goal of the motion stage of structure from motion is to estimate the absolute motions of the cameras starting from a set of matching points across the input images. Note that, in contrast to the case of multiple point-set registration, 3D points are not available, but only 2D (image) points. Let $M_{ij} \in SE(3)$ denote the rigid transformation between the reference frame of camera i and that of camera j, which is referred to as the *relative motion* of the pair (i, j), namely

$$M_{ij} = \begin{pmatrix} R_{ij} & \mathbf{t}_{ij} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \in SE(3) \tag{4.7}$$

with $R_{ij} \in SO(3)$ and $\mathbf{t}_{ij} \in \mathbb{R}^3$. As in the case of multiple point-set registration, the link between relative and absolute motions is given by Equation (4.5), which coincides with the consistency constraint of synchronization over $SE(3)$.

This implies that, *if* relative motions were known, the motion recovery stage of structure from motion would reduce to a rigid-motion synchronization, which in turn can be tackled directly as a synchronization over $SE(3)$, or by breaking the problem into rotation and translation and solving the two synchronization problems separately, according to the respective consistency definitions

$$R_{ij} = R_i R_j^\mathsf{T} \tag{4.8}$$

$$\mathbf{t}_{ij} = -R_i R_j^\mathsf{T} \mathbf{t}_j + \mathbf{t}_i \Longleftrightarrow \mathbf{z}_{ij} = \mathbf{x}_i - \mathbf{x}_j \tag{4.9}$$

where $\mathbf{x}_i = -R_i^\mathsf{T} \mathbf{t}_i$ represents the optical centre of camera i and $\mathbf{z}_{ij} = -R_i^\mathsf{T} \mathbf{t}_{ij}$ represents the baseline joining the centers of cameras i and j.

We now explain how relative motions can be computed in practice. The geometry of two views i and j is captured by the *essential matrix* E_{ij}, that is a 3×3 matrix satisfying the so-called *epipolar constraint*

$$\mathbf{p}_i^\mathsf{T} E_{ij} \mathbf{p}_j = 0 \tag{4.10}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ denote a pair of corresponding points between images i and j, expressed in (homogeneous) normalized coordinates, which are available if the interior parameters of the cameras are known. Note that (4.10) is a linear equation, thus the essential matrix can be estimated in closed-form if a sufficient number of corresponding points is known [87]. However, since the resulting system is homogeneous, such

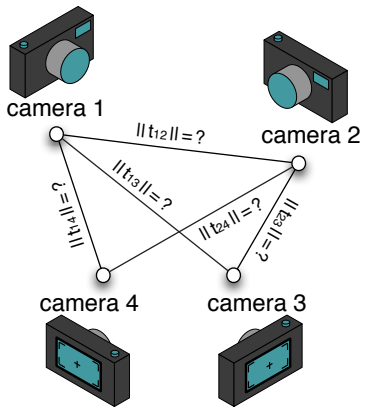


FIGURE 4.8: The epipolar graph.

a matrix is known up to scale. This fact is due to the depth-speed ambiguity which is inherent to the problem. The importance of the essential matrix lies in the following decomposition

$$E_{ij} = [\mathbf{t}_{ij}]_{\times} R_{ij} \tag{4.11}$$

which can be computed via singular value decomposition [88]. However, the scale indeterminacy in essential matrices transfers into relative translations. In other words, what can be computed from matching points are relative rotations R_{ij} and the *directions* of relative translations

$$\mathbf{v}_{ij} = \frac{\mathbf{t}_{ij}}{\|\mathbf{t}_{ij}\|} \tag{4.12}$$

but the translation magnitudes (also known as *epipolar scales*) $\alpha_{ij} = \|\mathbf{t}_{ij}\| = \|\mathbf{z}_{ij}\|$ are unknown (see Figure 4.8). The unit vector $\mathbf{v}_{ij}$ is also called the *relative bearing* of the pair (i, j). Thus, relative motions are not fully specified and hence structure from motion can not be straightforwardly solved as a rigid-motion synchronization.

There are three paths that can be followed in order to recover camera motion:

1. solve a rotation synchronization to obtain the angular attitudes of the cameras, then recover camera centers directly from the direction information (Section 4.4.2);

2. solve a rotation synchronization to obtain the angular attitudes of the cameras, then compute the epipolar scales, followed by a translation synchronization to recover camera centers (Section 4.4.3);

3. compute the epipolar scales and then solve a rigid-motion synchronization (Section 4.4.4).

4.4.2 Bearing-based Localization

As concerns the first possibility, which is detailed in Figure 4.9, note that the knowledge of absolute rotations (which are available after rotation synchronization) permits to consider the direction of the baselines instead of relative bearings

$$\mathbf{u}_{ij} = \frac{\mathbf{z}_{ij}}{\|\mathbf{z}_{ij}\|} = \frac{-R_i^{\mathsf{T}}\mathbf{t}_{ij}}{\|\mathbf{t}_{ij}\|}. \tag{4.13}$$

The unit vector $\mathbf{u}_{ij}$ is also called the *bearing* of the pair (i, j) and, in the absence of noise, it satisfies $\mathbf{u}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/\|\mathbf{x}_i - \mathbf{x}_j\|$. It can be viewed as the direction of the relative translation of the pair (i, j) expressed in an absolute reference frame.

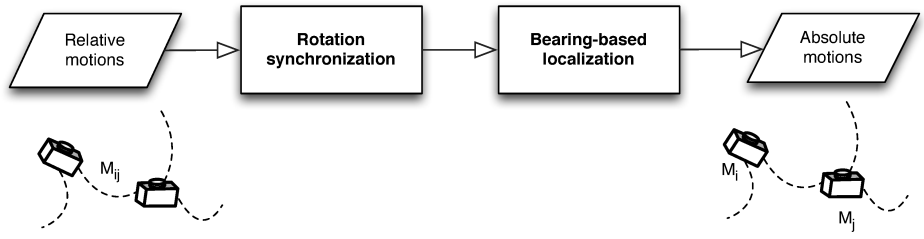


FIGURE 4.9: Camera motion estimation via rotation synchronization followed by bearing-based localization.

Thus recovering camera centers (positions) $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from pairwise directions $\mathbf{u}_{ij}$ is exactly an instance of *bearing-based network localization* in 3-space, which can be solved via the spectral method introduced in [33] (reviewed in Section 3.2.1), that follows a node-based approach. Other solutions include linear least squares [96], ℓ_∞ minimization [134], Riemannian gradient descent [181], the Levenberg-Marquardt algorithm [52, 195], semi-definite programming [141], quadratic programming [140], and the alternating direction method of multipliers [76]. See the survey in [142] for a detailed description of such techniques.

However, if bearing-based network localization is ill-posed, then any method will fail to produce a solution. The presence of ill-posed instances in practical scenarios (e.g., the case where the graph is not connected or all the constraints on a node are collinear) is also pointed out in [33] as a "problem pathology". Thus it is important to study the localizability of the problem before solving bearing-based localization. More precisely, the epipolar graph is required to be generically parallel rigid in order to guarantee that the problem is well-posed (see the theoretical analysis in Chapter 3).

4.4.3 Group Synchronization Pipeline

As concerns the second possibility, which is represented in Figure 4.10, note that, if rotations have been computed beforehand, then we are concerned with the problem

of estimating the epipolar scales $\alpha_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ starting from the knowledge of pairwise directions $\mathbf{u}_{ij}$, which coincides with solving bearing-based network localization via an edge-based approach. As explained in Section 3.2.2, if the graph is parallel rigid, the solution can be found as the 1-dimensional null-space of $C \odot U$, where C is a directed cycle basis of the epipolar graph and U contains all the directions in columns. In the presence of noise, Equation (3.26) is solved in the least-squares sense, by computing the least right singular vector of $C \odot U$.

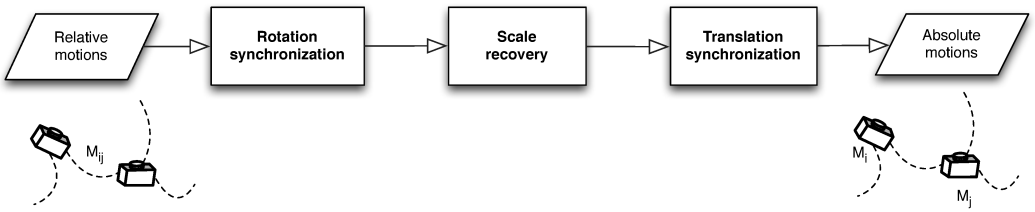


FIGURE 4.10: Camera motion estimation via rotation synchronization followed by translation synchronization.

Remark 4.1. This approach requires to compute m unknowns, representing edge lengths. If the graph is not sparse, the number of such unknowns is of the order $O(n^2)$, which is much larger than the node-based approach, which requires to recover $3n$ unknowns, representing node locations.

The previous remark motivates a divide-et-impera approach for computing the epipolar scales, which is at the basis of a scalable structure-from-motion pipeline, sketched in Figure 4.11. The pipeline – henceforth dubbed GSP, for Group Synchronization Pipeline – exploits the notion of group synchronization in most of its stages, which translates into direct solutions such as eigenvalue decompositions (rotations and scales) or linear least squares (translations).

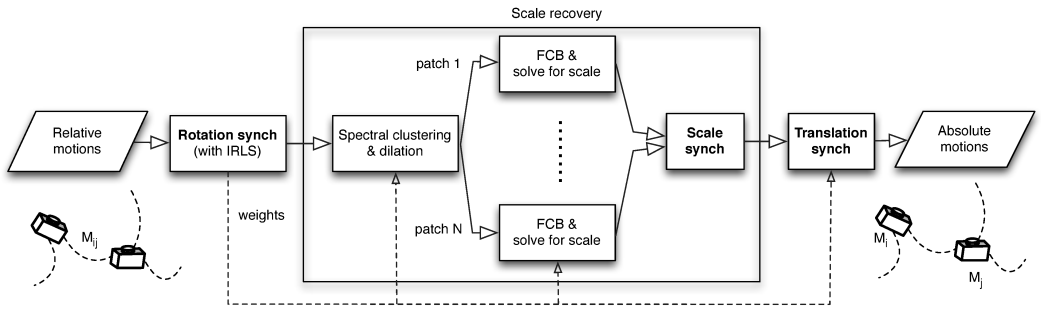


FIGURE 4.11: Overview of the Group Synchronization Pipeline. The group synchronization stages are highlighted in bold.

The first stage of GSP is rotation synchronization which is solved through the spectral approach detailed in Section 2.7, coupled with IRLS to gain resilience to outliers. Whereas in the IRLS iteration we use a "soft-redescender" – namely the Cauchy function (2.25), the final weights are computed with a "hard-redescender"

such as the bisquare function (2.26) that assigns zero weight to edges with residual higher than a threshold; in such a way outliers are definitively removed. A fixed threshold on the angular error (5° in our experiments) is also used as a safeguard against high outlier contamination. The final weights are attached to the edges of $\mathcal{G}$, and subsequently used throughout the pipeline wherever it makes sense.

In the second stage the epipolar scales are computed via a *divide-et-impera* approach: partition the graph, solve for the epipolar scales locally, and then globally synchronize all the scales. The idea of partitioning the computation into smaller sub-problems is also present, e.g., in [55] and [23].

The weighted epipolar graph is partitioned with spectral clustering, in particular using normalized cuts [165] that tend to produce clusters of approximately the same size. Such size is set equal to $\sqrt{n}$ so that the number of edges (and hence unknowns) in each cluster is of the order $O(n)$. In each cluster the largest parallel-rigid subgraph is extracted as explained in [103]. In order to obtain *overlapping* subgraphs, which are called *patches*, each cluster adopts some nodes of the other clusters, including orphan nodes left out by the parallel-rigid subgraph extraction. More precisely, the adoptable nodes must be connected to the cluster with at least two edges, in order to keep the rigidity of the patch (by the principle of triangulation). They are chosen among the ones with the highest degree and up to a fixed maximum number.

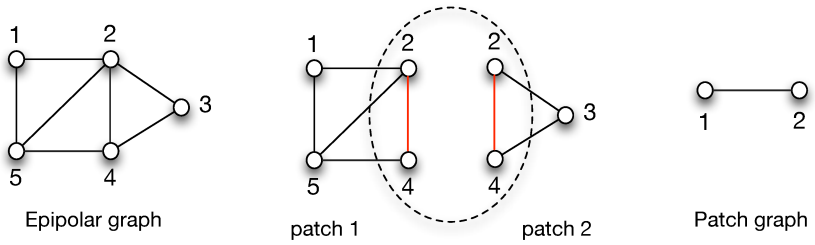


FIGURE 4.12: Patch graph.

In each patch the scales are computed as described in Section 3.2.2, using a maximum-weight spanning tree to obtain a fundamental cycle basis (FCB). This step can be performed in parallel, since each patch is independent from the others, thus speeding up the process.

Note that each patch has an unknown global scale $\beta_h \in \mathbb{R}$ and – thanks to the overlap among patches – such scales can be synchronized by running the spectral method described in Section 2.4 on the *patch graph* $\mathcal{G}^P$, whose vertices are the patches and two patches are adjacent if and only if they share at least one edge of the epipolar graph $\mathcal{G}$ (see Figure 4.12). Every such edge in common introduces a measure of the ratio $\beta_{hk} = \beta_h \beta_k^{-1}$ of the scales of the two adjacent patches: a single ratio is attached to the edge of $\mathcal{G}^P$ by least squares fitting.

Translation synchronization (as detailed in Section 2.3.2) is run in the end of GSP to compute the position of the cameras in the absolute reference system. The linear system of equations is solved via IRLS, initialized with the weights computed by rotation synchronization.

4.4.4 Scale Recovery from Relative Motions

The third possibility consists in recovering the unknown magnitudes of relative translations between camera pairs based on the knowledge of *relative* rotations and translation directions and, as it will be shown, it is tightly related to the edge-based formulation of bearing-based localization. Specifically, the goal is to reduce all such unknowns into a single scale, which cannot be eliminated. This allows to recover camera motion via rigid-motion synchronization, as shown in Figure 4.13.

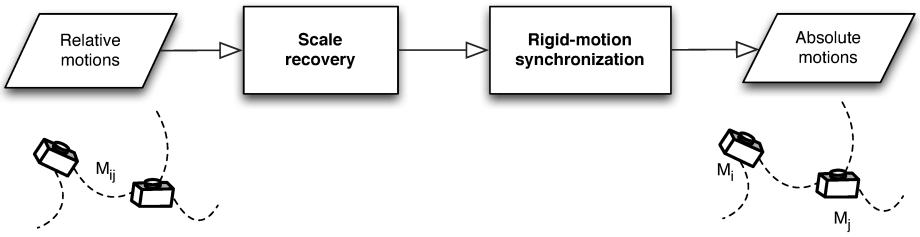


FIGURE 4.13: Camera motion estimation via rigid-motion synchronization.

Let us consider the null-cycle constraint over $SE(3)$. If $\{(1,2),(2,3),\ldots,(\ell-1,\ell)\}$ denotes a circuit of length $\ell \geq 3$ in the epipolar graph, then the composition of pairwise motions along such circuit must return the identity, namely

$$M_{12}M_{23}\ldots M_{\ell-1,\ell}M_{\ell 1} = I_4. \tag{4.14}$$

Equation (4.14) is called the *compatibility constraint*, and it can also be expressed as

$$M_{12}M_{23}\ldots M_{\ell-1,\ell} = M_{1\ell} \tag{4.15}$$

which, considering separately the rotation and translation terms, results in

$$R_{12}R_{23}\ldots R_{\ell-1,\ell} = R_{1\ell} \tag{4.16}$$

$$\mathbf{t}_{12} + \sum_{k=2}^{\ell-1}\Big(\prod_{i=1}^{k-1} R_{i,i+1}\Big)\mathbf{t}_{k,k+1} = \mathbf{t}_{1\ell} \tag{4.17}$$

where the relation between translations can be rewritten in terms of relative bearings and epipolar scales

$$\alpha_{12}\mathbf{v}_{12} + \sum_{k=2}^{\ell-1}\Big(\prod_{i=1}^{k-1} R_{i,i+1}\Big)\alpha_{k,k+1}\mathbf{v}_{k,k+1} = \alpha_{1\ell}\mathbf{v}_{1\ell} \tag{4.18}$$

which is a homogeneous linear equation in the unknown scales.

In a general epipolar graph, we can collect the compatibility constraints coming from a (directed) cycle basis, resulting in a linear system of the form

$$F\boldsymbol{\alpha} = \mathbf{0} \tag{4.19}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^m$ is the stack of the epipolar scales and the entries of F depend on the relative rotations and translation directions, according to (4.18). Each triplet of rows in F corresponds to a circuit, and each column corresponds to an edge in the epipolar graph, resulting in a matrix of dimension $3(m-n+1) \times m$. This suggests a two-stage method to recover the unknown translation magnitudes: first, a cycle basis for the epipolar graph is computed; then all the epipolar scales are recovered simultaneously by solving a homogeneous linear system, where the solution is unique (up to scale) if and only if $\dim(\ker(F)) = 1$, or, equivalently, if and only if $\operatorname{rank}(F) = m-1$. In the presence of noise, the solution is found by taking the least right singular vector of F, which corresponds to solving (4.19) in the least-squares sense.

Zeller-Faugeras method

Our method for computing the epipolar scales can be seen as an extension of the Zeller-Faugeras method [199], which is briefly reviewed here, to general epipolar graphs. The authors of [199] consider a *sequence* of n images, whose epipolar graph is represented in Figure 4.14, where the following compositional rule holds

$$\mathbf{t}_{1i} = R_{12}\mathbf{t}_{2i} + \mathbf{t}_{12} \tag{4.20}$$

which is equivalent to

$$\alpha_{1i}\mathbf{v}_{1i} = \alpha_{2i}R_{12}\mathbf{v}_{2i} + \alpha_{12}\mathbf{v}_{12}. \tag{4.21}$$

This leads to the following solution for the ratios of the epipolar scales

$$\frac{\alpha_{12}}{\alpha_{1i}} = \frac{(R_{12}\mathbf{v}_{2i} \times \mathbf{v}_{1i})^{\mathsf{T}}(R_{12}\mathbf{v}_{2i} \times \mathbf{v}_{12})}{\|R_{12}\mathbf{v}_{2i} \times \mathbf{v}_{12}\|^2}. \tag{4.22}$$

More precisely, if we arbitrarily fix the value of (e.g.) α_{12}, then we can compute the remaining scales α_{1i} by using the equations above. The arbitrary choice of α_{12} corresponds to the global scale indeterminacy, which can not be avoided without external measurements.

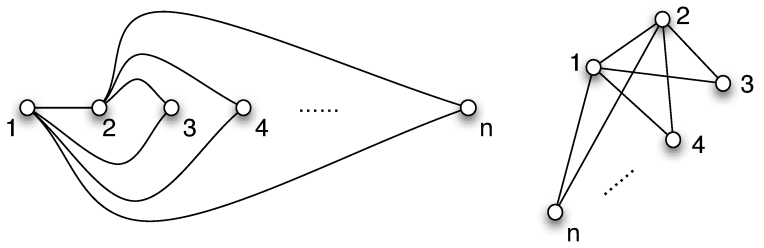


FIGURE 4.14: The epipolar graph corresponding to the Zeller-Faugeras method [199]. It is made of $n-2$ circuits of length 3 all sharing a common edge.

Local versus global frames

We now explain how our method for computing the epipolar scales from relative bearings relates to the edge-based formulation of bearing-based localization. Let us assume that the absolute rotations of the cameras are known. Using the consistency constraint over $SO(3)$, that is $R_{ij} = R_i R_j^\mathsf{T}$, all the factors in (4.18) simplify except of the first and the last one, thus the product of relative rotations in (4.18) reduces to $R_1 R_k^\mathsf{T}$. By multiplying both sides by $-R_1^\mathsf{T}$, we obtain

$$-\alpha_{12} R_1^\mathsf{T} \mathbf{v}_{12} - \sum_{k=2}^{\ell-1} \alpha_{k,k+1} R_k^\mathsf{T} \mathbf{v}_{k,k+1} = -\alpha_{1\ell} R_1^\mathsf{T} \mathbf{v}_{1\ell} \tag{4.23}$$

which coincides with

$$\sum_{k=1}^{\ell-1} \alpha_{k,k+1} \mathbf{u}_{k,k+1} = \alpha_{1\ell} \mathbf{u}_{1\ell} \tag{4.24}$$

since the baseline directions are related to the relative translations through the formula $\mathbf{u}_{ij} = -R_i^T \mathbf{v}_{ij}$. Note that Equation (4.24) coincides with (3.52), which, if the equations coming from a cycle basis are stacked, becomes $(C \odot U)\boldsymbol{\alpha} = \mathbf{0}$, where C denotes the cycle matrix, thus yielding the edge-based formulation of bearing-based localization. Accordingly, the unknown translation magnitudes can be uniquely recovered (up to scale) if and only if the epipolar graph is parallel rigid.

Please observe that the matrix $C \odot U$ is *not* equal to the matrix F used in (4.19), but it has the same size and the same null-space (in the noise-free case). Each row in F is of the form

$$-R_k(\mathbf{c}_k^\mathsf{T} \odot U)\ \boldsymbol{\alpha} = (\mathbf{c}_k^\mathsf{T} \odot -R_k U)\ \boldsymbol{\alpha} = \mathbf{0} \tag{4.25}$$

where R_k is a rotation that takes into account the fact that in each circuit $\mathbf{c}_k$ an arbitrary local reference system has been considered. Hence, there exists a choice of rotations $R_1, \ldots, R_m$ such that

$$\begin{bmatrix} -R_1(\mathbf{c}_1^\mathsf{T} \odot U) \\ \vdots \\ -R_m(\mathbf{c}_m^\mathsf{T} \odot U) \end{bmatrix} = \begin{bmatrix} -R_1 & & \\ & \ddots & \\ & & -R_m \end{bmatrix} (C \odot U) = F. \tag{4.26}$$

In summary, the equations involving the bearings and those involving the relative motions are equivalent in terms of constraints on the solution, however they configure two different approaches: the bearings in (4.24) require to compute the absolute rotations *before* the epipolar scales, whereas the equations in (4.18) are written with respect to independent local frames, thereby avoiding the need to solve for the absolute rotations beforehand. The former yield a more compact matrix formulation which simplifies the discussion, as done in Chapter 3.

4.5 Conclusion

In this chapter we showed how the framework of synchronization can be profitably used in several Computer Vision applications, including multi-view matching, multiple point-set registration and structure from motion, and we pointed out the link between structure from motion and bearing-based network localization. We also proposed a divide and conquer technique for computing the epipolar scales in a structure from motion pipeline that exploits a synchronization instance. Future work will explore techniques for solving partitioned linear systems (e.g. [35]) as an alternative way to divide scale recovery into smaller subproblems.

Chapter 5

Synthetic Experiments

In this chapter we report experiments on synthetic data in oder to evaluate the proposed solutions against state-of-the art algorithms. In particular, we consider the synchronization problem over $ISym(d)$ (Section 5.1), over $SO(3)$ (Section 5.2) and over $SE(3)$ (Section 5.3). All the experiments were performed in Matlab on a dual-core MacBook Air with i5 1.3GHz processor and 4Gb RAM.

5.1 Synchronization over $ISym(d)$

We compared our solution to synchronization over $ISym(d)$ – detailed in Section 2.10 and henceforth dubbed PARTIALSYNCH – to the method in [143] (which will be referred to as TOTALSYNCH[1]). Performances were measured in terms of *precision* (number of correct matches returned divided by the number of matches returned) and *recall* (number of correct matches returned divided by the number of correct matches that should have been returned). In order to provide a single figure of merit we computed the *F-score* (twice the product of precision and recall divided by their sum), which is a measure of accuracy and reaches its best value at 1.

In our simulations a fixed number of $d = 20$ objects was chosen, while the number of nodes varied from $n = 10$ to $n = 50$. The *observation ratio*, i.e., the probability that an object is seen in a node, decreased from 1 (that corresponds to total permutations) to 0.2. After generating ground-truth absolute permutations, pairwise matches were computed from Equation (4.1). Then random errors were added to relative permutations by switching two matches, removing true matches or adding false ones. The *input error rate*, i.e., the ratio of mismatches, varied from 0 to 0.8. For each configuration the test was run 20 times and the mean F-score was computed. In order to evaluate a solution, the total permutation that best aligns the estimated absolute permutations onto ground-truth ones was computed with the Kuhn-Munkres algorithm [110].

Results are reported in Figure 5.1, which shows the F-score for the two methods as a function of number of nodes, observation ratio and input error rate. In the case of total permutations (observation ratio = 1) both techniques perform well. Our method (PARTIALSYNCH) correctly recovers the absolute permutations even when not all

[1]The code is available at `http://pages.cs.wisc.edu/~pachauri/perm-sync/`

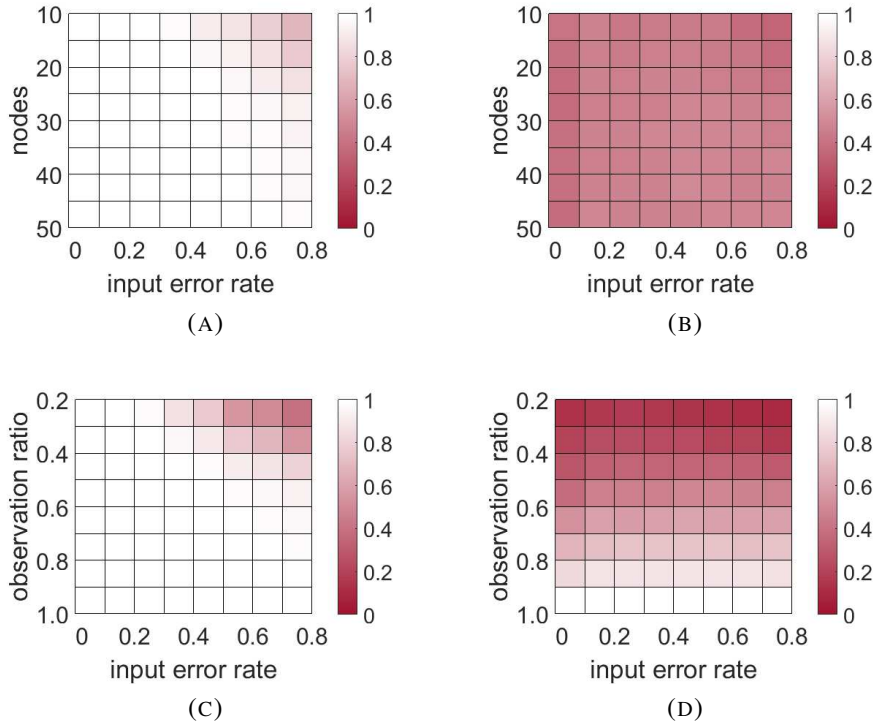


FIGURE 5.1: F-score (the higher the better) of PARTIALSYNCH (a,c) and TOTALSYNCH (b,d). In (a,b) the number of nodes n and the input error rate are varying, whereas the observation ratio is constant and equal to 0.6. In (c,d) the observation ratio and input error rate vary with $n = 30$.

the objects are seen in every node, and in the presence of high contamination. On the contrary, TOTALSYNCH cannot deal with partial permutations, indeed its performances degrade quickly as the observation ratio decreases. In general, the accuracy increases with the number of nodes.

5.2 Synchronization over $SO(3)$

We evaluated the low-rank solution to synchronization over $SO(3)$, which is detailed in Section 2.5.5, in terms of accuracy, execution cost and robustness to outliers. More precisely, we plugged R-GODEC (Algorithm 3), GRASTA [90] and L1-ALM [202] in our framework, obtaining three rotation synchronization methods based on LRS matrix decomposition, which were compared to several techniques from the state of the art.

We considered the spectral solution (EIG) [4], the semidefinite relaxation (SDP) [4], the rank relaxation (OPTSPACE) [105], the Weiszfeld algorithm [85], the L1-IRLS algorithm [44], and the LUD algorithm [187]. We also included in the comparison the spectral solution coupled with IRLS, dubbed EIG-IRLS. The null-space method is not considered here since, as it will be shown in the Section 5.3, it is comparable in accuracy to EIG while being slower. The code of LUD was provided by

the authors of [187], the codes of GRASTA, L1-ALM, OPTSPACE and L1-IRLS are available on the web, while in the other cases we used our implementation[2]. The SeDuMi toolbox [172] was used to solve the semidefinite program associated to the SDP method. All the methods used the default tuning parameter(s) specified in the original paper or code. In particular, for R-GODEC the value of λ was computed by plugging $\sigma = 0.02$ in formula (C.7).

In order to compare estimated and ground-truth absolute rotations we employed ℓ_1 single averaging. Specifically, if $\widehat{R}_1, \ldots, \widehat{R}_n$ are estimates of the theoretical absolute rotations $R_1, \ldots, R_n$, then the optimal $S \in SO(3)$ that aligns them into a common reference system solves $R_i = \widehat{R}_i S$, and hence it is the single mean of the set $\{R_i \widehat{R}_i^T, i = 1, \ldots, n\}$, and it can be computed e.g. by using [85]. Then we used the angular distance to evaluate the accuracy of rotation recovery. The angular (or geodesic) distance between two rotations A and B is the angle of the rotation BA^T (in the angle-axis representation) so chosen to lie in the range $[0, 180^\circ]$, namely $d_\angle(A,B) = d_\angle(BA^T, I) = 1/\sqrt{2}\left\|\log(BA^T)\right\|_2$. Other distances in SO(3) can be considered with comparable results.

In our simulations we considered n rotation matrices sampled from random Euler angles, representing ground truth absolute rotations. The measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a random graph drawn from the Erdős-Rényi model with parameters (n, p), i.e. given a vertex set $\mathcal{V} = \{1, 2, \ldots, n\}$ each edge (i, j) is in the set $\mathcal{E}$ with probability $p \in [0, 1]$, independently of all other edges. Thus $(1 - p)$ controls the degree of sparsity of the graph and $p = 1$ corresponds to the complete graph. Only connected graphs are considered among all the instances generated in this way. A fraction of the pairwise rotations was drawn uniformly from SO(3), simulating outliers. The remaining pairwise rotations were corrupted by multiplicative noise $\widehat{R}_{ij} = R_{ij} N_{ij}$ where $N_{ij} \in$ SO(3) has axis uniformly distributed over the unit sphere and angle following a Gaussian distribution with zero mean and standard deviation $\sigma_R \in [1^\circ, 10^\circ]$, thus representing a small perturbation of the identity matrix. All the results were averaged over 50 trials.

It is hard to evaluate the performances of a synchronization method as a whole, since several factors are involved, thus in the following simulations we let one parameter vary at a time and keep the others fixed.

Noise

In this experiment we analyze the behavior of the aforementioned methods in the presence of noise among the input rotations without introducing outliers, with $n = 100$. Results are reported in Figure 5.2 with $p = 0.5$ and $p = 0.2$, which correspond to about 50% and 80% of missing pairs, respectively. As expected the lowest errors are achieved by non-robust methods, namely EIG, SDP and OPTSPACE. On the contrary, all the robust methods yield worse results, since they essentially trade

[2]The code of R-GODEC is available at `http://www.diegm.uniud.it/fusiello/demo/gmf/`

robustness for statistical efficiency. This compromise is particularly evident in L1-IRLS at the point when it switches from quadratic to fixed loss, defined by a fixed value (5°), whereas EIG-IRLS, that uses a data dependent threshold, has a more linear trend, for the trade-off takes place at all noise levels.

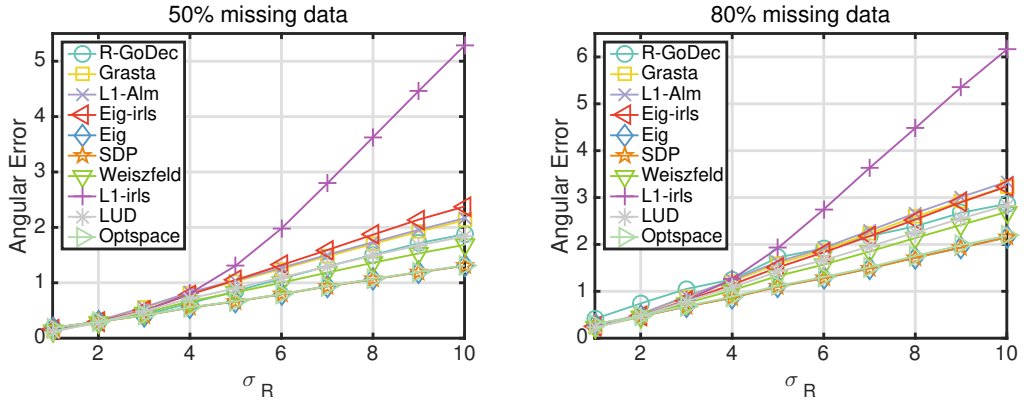


FIGURE 5.2: Mean angular errors [degrees] as a function of the noise standard deviation, with $p = 0.5$ (left) and $p = 0.2$ (right). Outliers are not introduced in this experiment.

Outliers

In this experiment we study the robustness to outliers of our approach. Each edge $(i, j) \in \mathcal{E}$ was designated as an outlier with uniform probability $q \in [0, 1]$, independently of all other edges. Figure 5.3 shows the angular errors of all the analyzed methods as a function of q, with $n = 100$. As before, we chose $p = 0.2$ and $p = 0.5$ to define the density of the measurement graph. In this experiment all the inlier rotations were corrupted by a fixed level of noise ($\sigma_R = 5°$). When the percentage of unspecified relative rotations is about 50%, the errors of R-GODEC, GRASTA and L1-ALM remain almost constant, showing no sensitivity to outliers. The same happens for LUD, L1-IRLS and EIG-IRLS. On the contrary, EIG, SDP and OPTSPACE are not robust to outliers.As for the Weiszfeld algorithm, its performances places it at the middle between robust and non-robust solutions. Specifically, it shows good resilience to outlier rotations when they are below 30%, then the errors start to grow up, yielding a behaviour similar to non-robust approaches. When the data matrix is highly incomplete ($p = 0.2$), the difference between robust and non-robust solutions becomes smaller, however results are qualitatively similar to the previous case.

Missing Data

In this experiment we study how missing data influence the performances of LRS algorithms. Figure 5.4 reports the angular errors of the analysed methods as a function of $(1 - p)$, with $n = 100$. The sparsity parameter $(1 - p)$ ranges from 0.5 to 0.95, which correspond to about 50% and 95% of missing pairs. Results with lower values of $(1 - p)$ yield the same behaviour as $(1 - p) = 0.5$, and hence they are not

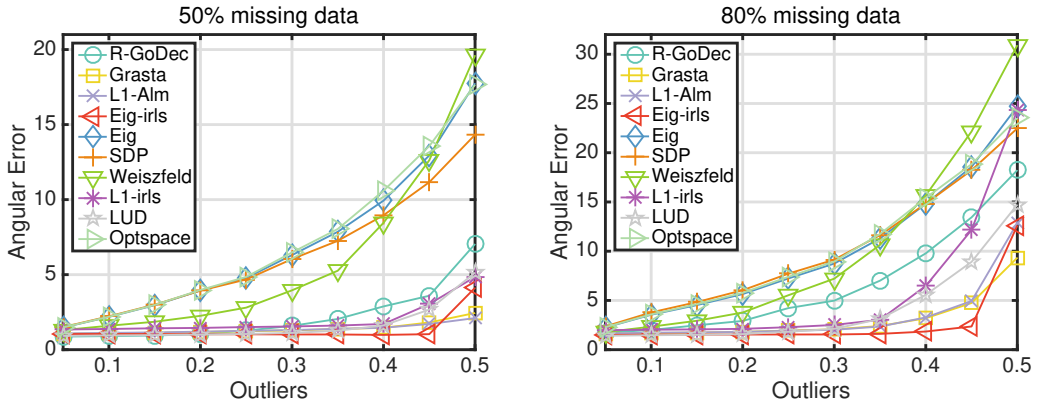


FIGURE 5.3: Mean angular errors [degrees] as a function of q, with $p = 0.5$ (left) and $p = 0.2$ (right). A fixed level of noise is applied to the inlier rotations in this experiment.

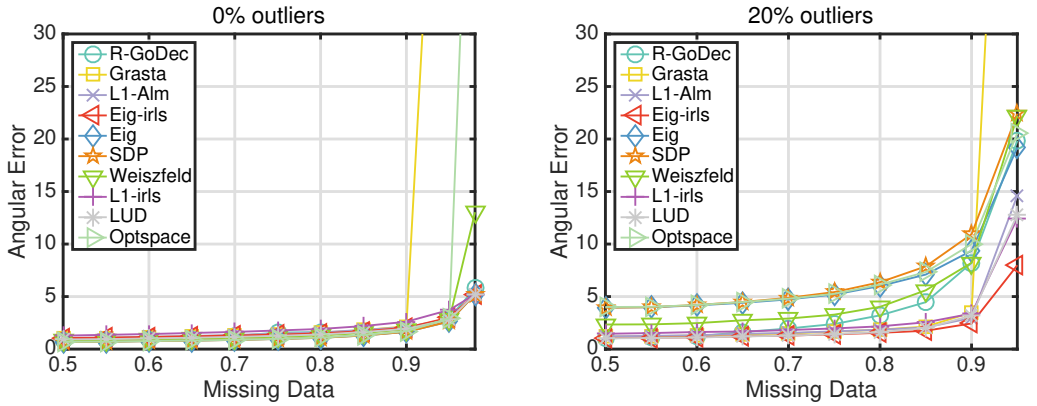


FIGURE 5.4: Mean angular errors [degrees] as a function of $(1-p)$, with $q = 0$ (left) and $q = 0.2$ (right). A fixed level of noise is applied to the inlier rotations in this experiment. The average angular error of GRASTA is approximately 80° for $(1-p) = 0.95$.

reported. We considered both the ideal case where outliers are not present ($q = 0$) and a more realistic situation in which a given percentage of outliers is introduced ($q = 0.2$). In the first case we also considered the minimal situation in which $n-1$ relative rotations are available, which corresponds to 98% of missing pairs. In both cases all the inlier rotations were corrupted by a fixed level of noise ($\sigma_R = 5^\circ$). In the absence of outliers, when the percentage of missing pairs do not exceed 90%, the errors obtained by our approach and the remaining methods remain constant, showing no sensitivity to missing data. GRASTA can tolerate up to 90% of missing pairs, whereas R-GODEC and L1-ALM can also handle the minimal situation, but the errors are higher then the previous cases. Indeed, there is no way to compensate the initial errors since there is no redundancy. However, it should be noted that if only $n-1$ relative rotations are available, then there is no need to perform rotation synchronization, and the absolute rotations can be computed by propagating the consistency constraint (2.6) along a spanning tree, starting from any node assumed equal to the identity matrix. In the presence of outliers, GRASTA can tolerate up to 90% of missing data, while R-GODEC and L1-ALM give reasonable results until 95% of missing data. However, the performances of R-GODEC degrade starting from 80%

of missing pairs, reaching errors comparable to non-robust methods when the percentage of missing pairs is 90%. As for the remaining algorithms, their behaviour is qualitatively similar to the case where outliers are not present.

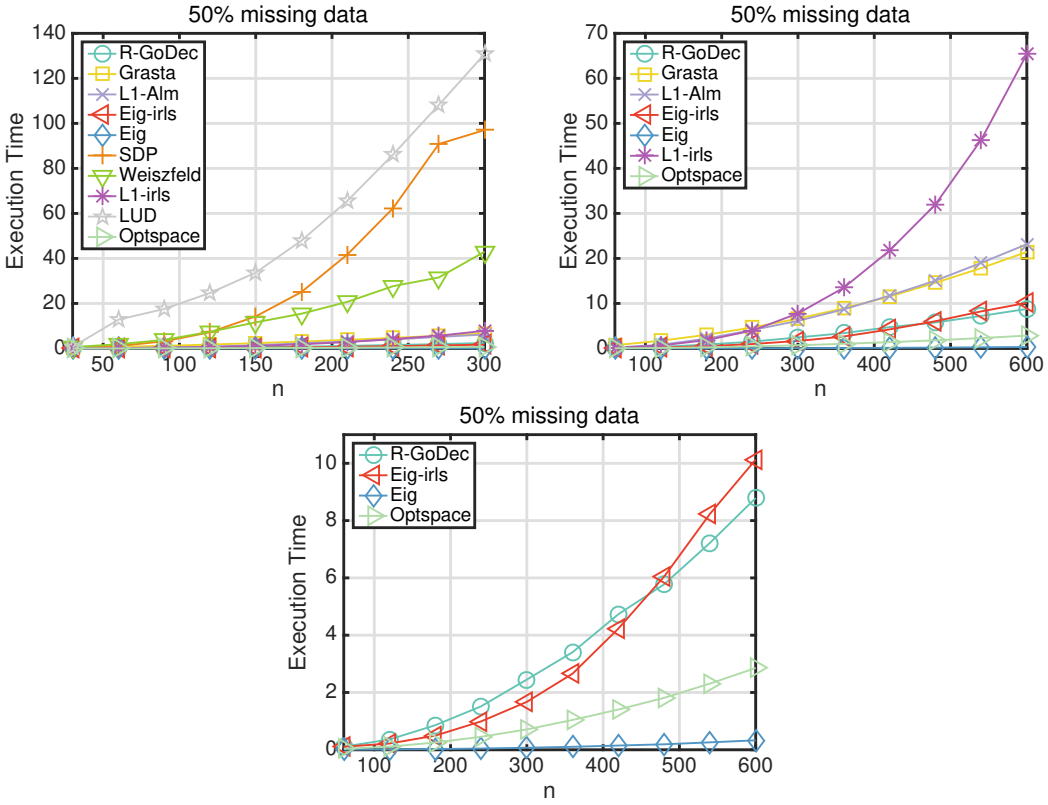


FIGURE 5.5: Execution times [seconds] as a function of the number of absolute rotations, with $p = 0.5$ and $q = 0.2$. A fixed level of noise is applied to the inlier rotations in this experiment. LUD, SDP and Weiszfeld are analysed only with a maximum of 300 nodes due to computational limitations. The top-right and bottom figures are a magnification of the top-left one.

Execution Time

In this experiment we analyse the computational efficiency of our approach in two situations. First, we kept the density level of the measurement graph fixed ($p = 0.5$) and let n vary between 30 and 600. Then, we kept the number of absolute rotations fixed ($n = 100$) and let $(1 - p)$ vary between 0.05 (about 5% of missing data) and 0.95 (about 95% of missing data). In both cases we introduced a fixed level of noise and outliers on relative rotations ($\sigma_R = 5°$, $q = 0.2$). Figure 5.5 shows that LUD, SDP and Weiszfeld qualify as the slowest algorithms, while the other ones are significantly faster. In particular, the EIG method is the fastest solution to the rotation synchronization problem, but it is not robust. Among all the robust methods, R-GODEC and EIG-IRLS achieve the lowest execution times, outperforming L1-IRLS. The execution times of GRASTA and L1-ALM are slightly higher than R-GODEC. Figure 5.6 shows that LUD and Weiszfeld require more time as the viewing

graph gets denser, whereas the execution times of the other techniques do not change significantly as p varies.

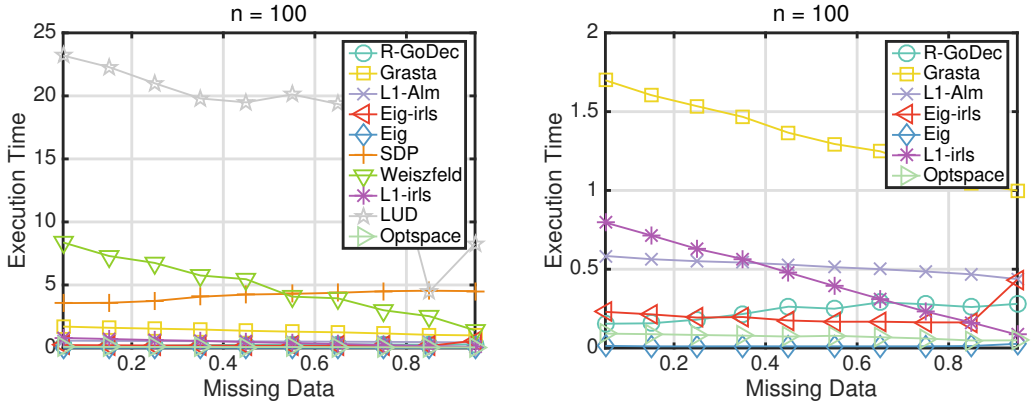


FIGURE 5.6: Execution times [seconds] as a function of $(1-p)$, with $n = 100$ and $q = 0.2$. A fixed level of noise is applied to the inlier rotations in this experiment. The right figure is a magnification of the left one.

Outlier Detection

We conclude this analysis by discussing the performances of our approach in terms of outlier detection, although this is not strictly part of rotation synchronization. While R-GODEC and GRASTA separate outliers and noise, the L1-ALM algorithm does not perform a classification of the data into inliers/outliers, thus it is not considered in this experiment. Outliers correspond to non-zero entries in the sparse term, namely S_1 for R-GODEC and S for GRASTA, with reference to Equations (C.21) and (C.15), respectively. We considered the receiver operating characteristic (ROC) curve, where the x-axis is the fraction of inliers erroneously classified as outliers, and the y-axis is the fraction of outliers correctly detected. The parameters that balance sparsity of outliers and noise are λ for R-GODEC and ρ for GRASTA (see Equation (C.17)), thus each point in the ROC space is associated to a specific value of λ or ρ, respectively. Figures 5.7 and 5.8 show the ROC curves of R-GODEC and GRASTA for different percentages of missing data and outliers, with $n = 100$. All the inlier rotations were corrupted by a fixed level of noise ($\sigma_R = 5°$). With reference to Figure 5.7, it is remarkable that R-GODEC gives a perfect classification with up to 50% of outliers. The performances drop with 60% of outliers, which is however a fairly high degree of contamination. Figure 5.8 evidences that R-GODEC is generally more accurate than GRASTA in terms of classification, probably thanks to the mixed $\ell_{2,1}$-norm that promotes a block structure in S_1.

5.3 Synchronization over $SE(3)$

We evaluated the LRS formulation detailed in Section 2.5.5 and the spectral/null-space solutions proposed in Section 2.8 for synchronization over SE(3), analyzing

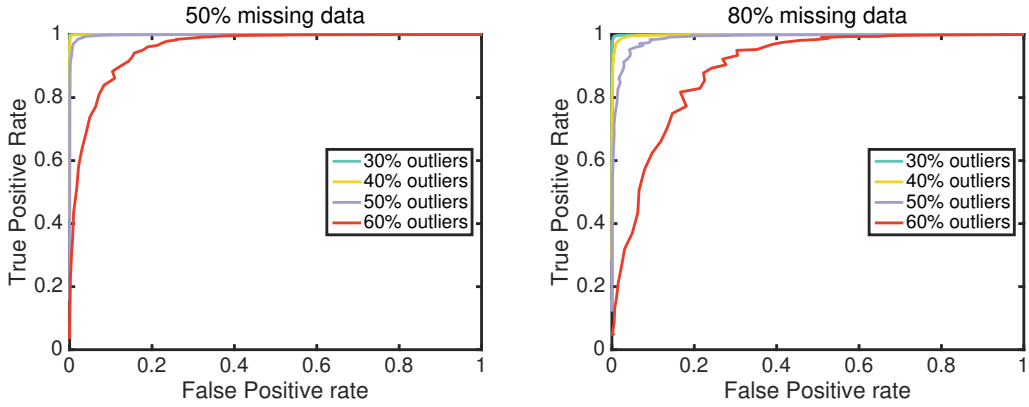


FIGURE 5.7: ROC curves of outlier detection for R-GODEC, with $p = 0.5$ (left) and $p = 0.2$ (right). A fixed level of noise is applied to the inlier rotations in this experiment.

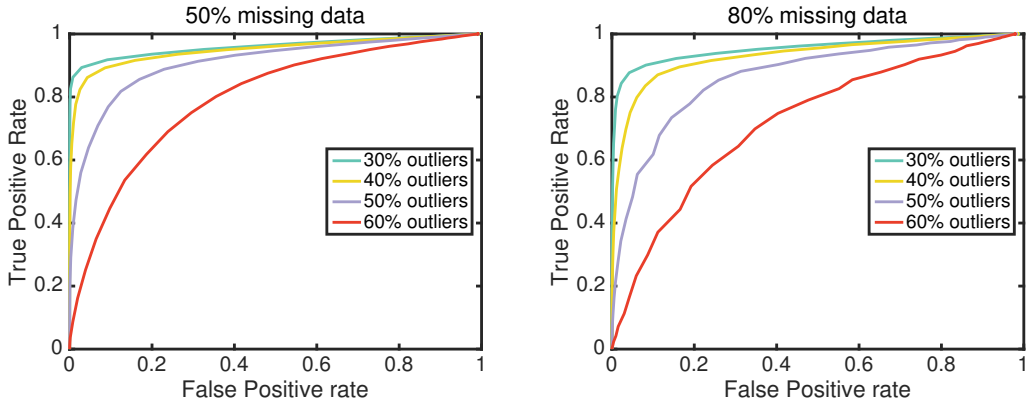


FIGURE 5.8: ROC curves of outlier detection for GRASTA, with $p = 0.5$ (left) and $p = 0.2$ (right). A fixed level of noise is applied to the inlier rotations in this experiment.

resilience to noise, robustness to outliers, sensitivity to missing data and computational cost. The null-space method is named Null-SE(3), the spectral solution is dubbed EIG-SE(3) and its robust variation is called EIG-SE(3)-IRLS. As concerns IRLS, we defined the residual in terms of the rotational component only, since it is reasonable to assume that if a relative motion is an outlier, then both the rotation and translation components are wrong. As done in Section 5.2, we considered three LRS decomposition algorithms, namely R-GODEC, GRASTA and L1-ALM.

We compared such algorithms to several techniques from the state of the art, namely the methods developed by Sharp et al. [163], Govindu [79], Torsello et al. [177] (DIFFUSION), and Rosen et al. [150]. The codes of GRASTA, L1-ALM, DIFFUSION and Rosen et al. are available online, the one by Govindu was provided by the author, while in the other cases we used our implementation[3]. All the methods used the default tuning parameter(s) specified in the original paper or code. In particular, for R-GODEC the value of λ was computed by plugging $\sigma = 0.02$ in formula (C.7).

[3]The codes of EIG-SE(3) and Null-SE(3) are available at `http://www.diegm.uniud.it/fusiello/demo/gmf/`

In order to compare estimated and ground-truth absolute motions, we found the optimal isometry that aligns them by applying single averaging for the rotation term and least-squares for the translation term. Specifically, if $\widehat{M}_1, \ldots, \widehat{M}_n$ are estimates of the theoretical absolute motions $M_1, \ldots, M_n$ then the optimal $N \in SE(3)$ that aligns them into a common reference system solves $M_i = \widehat{M}_i N$, which is equivalent to $R_i = \widehat{R}_i R$ and $\mathbf{t}_i = \widehat{R}_i \mathbf{t} + \widehat{\mathbf{t}}_\mathbf{i}$ by considering separately the rotation and translation term. Thus the optimal $R \in SO(3)$ is the single mean of the set $\{R_i \widehat{R}_i^\mathsf{T}, i = 1, \ldots, n\}$, which can be estimated by applying ℓ_1 single averaging [85], while the optimal $\mathbf{t} \in \mathbb{R}^3$ is computed in the least-squares sense. Then we used the angular distance and Euclidean norm to measure the accuracy of estimated rotations and translations respectively,

We considered n absolute motions in which rotations were sampled from random Euler angles and translation coordinates followed a standard Gaussian distribution. As done in Section 5.2, the measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ was drawn from the Erdős-Rényi distribution with parameters (n, p), and it was discarded if not connected. The inlier pairwise motions were corrupted by a multiplicative noise $\widehat{M}_{ij} = M_{ij} E_{ij}$, with $E_{ij} \in$ SE(3) representing a small perturbation of the identity matrix. The rotation component of E_{ij} was generated as in Section 5.2, and the translation component was sampled from a Gaussian distribution with zero mean and standard deviation $\sigma_T \in [0.01, 0.1]$. All the results were averaged over 50 trials.

Spectral solution versus null-space solution

We first compare the spectral and null-space solutions through two experiments. In the first simulation we evaluate the effect of noise on relative motions in the absence of outliers, considering $n = 100$ absolute motions and $p = 0.5$, which corresponds to about 50% of missing pairs. Figure 5.9 reports the mean errors on absolute motions (rotation errors are measured in degrees while translation errors are commensurate with the simulated data) as a function of the standard deviation of noise, showing that there are no significant differences between EIG-SE(3) and Null-SE(3). In another experiment we studied the computational efficiency of the two methods, considering $p = 0.5$ and letting n vary between 60 and 600. We introduced a fixed level of noise on relative motions ($\sigma_T = 0.05, \sigma_R = 5^\circ$). Figure 5.10 shows that EIG-SE(3) is faster then Null-SE(3), therefore we drop Null-SE(3) in subsequent comparisons.

Noise

In this experiment we evaluate the effect of noise on relative motions in the absence of outliers, with $n = 100$ and $p = 0.5$. Figure 5.11 shows the mean errors on absolute motions obtained by all the analysed techniques, as a function of the standard deviation of noise. The worst resilience to noise is achieved by Sharp et al. whereas the remaining algorithms return good estimates of absolute motions. A possible explanation of such behaviour is that in [163] the error is distributed among the motions but it

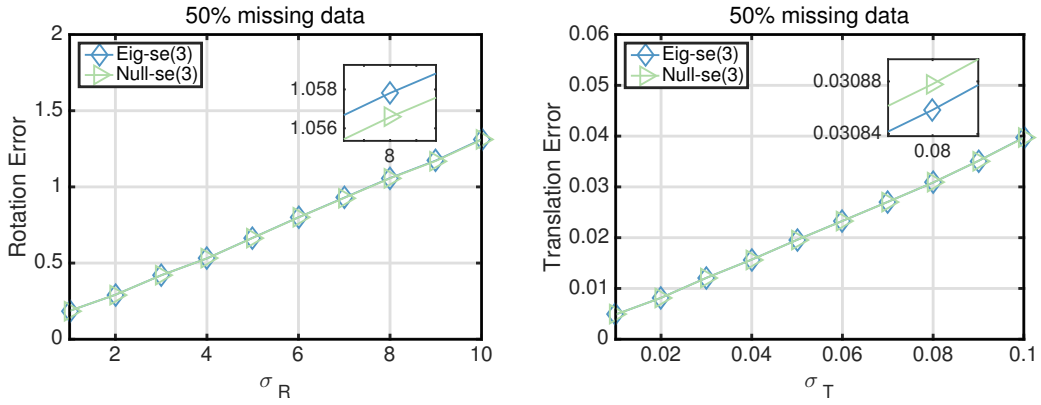


FIGURE 5.9: Mean errors on absolute motions as a function of the noise standard deviation, with $p = 0.5$. Outliers are not introduced in this experiment.

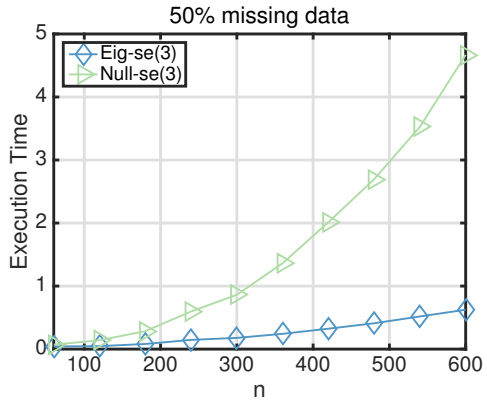


FIGURE 5.10: Execution times [seconds] as a function of n, with $p = 0.5$. Outliers are not introduced and a fixed level of noise is applied to the inlier relative motions in this experiment.

is not reduced. The best accuracy is achieved by non robust methods – namely EIG-SE(3), DIFFUSION, Govindu and Rosen et al., whereas robust techniques – namely EIG-SE(3)-IRLS, R-GODEC, GRASTA and L1-ALM, trade robustness for statistical efficiency.

Outliers

In this experiment we study the robustness to outliers of the proposed solutions. Each edge $(i, j) \in \mathcal{E}$ was designated as an outlier with uniform probability $q \in [0, 1]$, independently of all other edges. Outlier edges were assigned random elements of SE(3). We considered $n = 100$ absolute motions sampled as before, we chose $p = 0.5$ to define the density of the measurement graph, and we introduced a fixed level of noise on relative motions ($\sigma_T = 0.05, \sigma_R = 5°$). The probability q that an edge is outlier ranges from 0.05 to 0.5, which correspond to about 5% and 50% of effective outliers. Results are reported in Figure 5.12, which shows the mean errors on absolute motions as a function of q. The errors obtained by Sharp et al. are not reported so as to better visualize differences between the remaining algorithms (the method in [163] yields

an average rotation error of 20° for $q = 0.05$ and 100° for $q = 0.5$). Figure 5.12 confirms that EIG-SE(3), DIFFUSION, the methods by Govindu and Rosen et al. are not robust, and it clearly shows the resilience to outliers gained by R-GODEC, GRASTA, L1-ALM and EIG-SE(3)-IRLS. In particular, the errors obtained by LRS decomposition techniques remain almost unchanged until $q = 0.45$ for rotations and $q = 0.3$ for translations, whereas the breakdown point of EIG-SE(3)-IRLS is $q = 0.45$ for both rotations and translations.

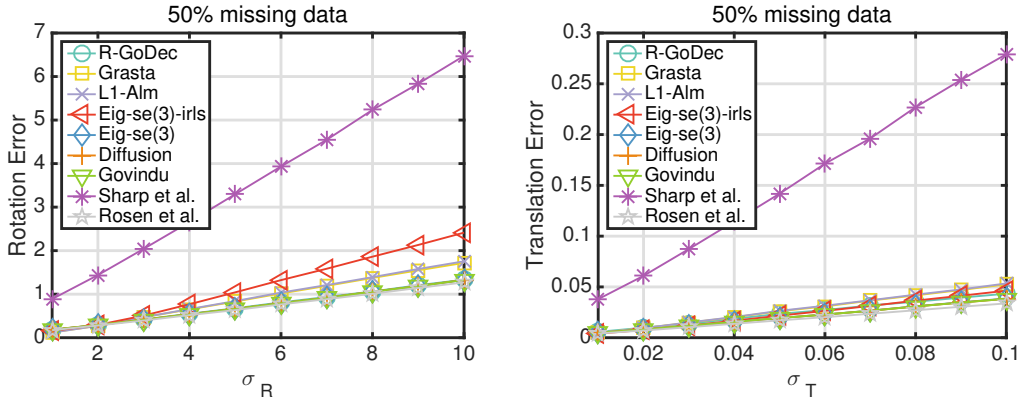


FIGURE 5.11: Mean errors on absolute motions as a function of the noise standard deviation, with $p = 0.5$. Outliers are not introduced in this experiment.

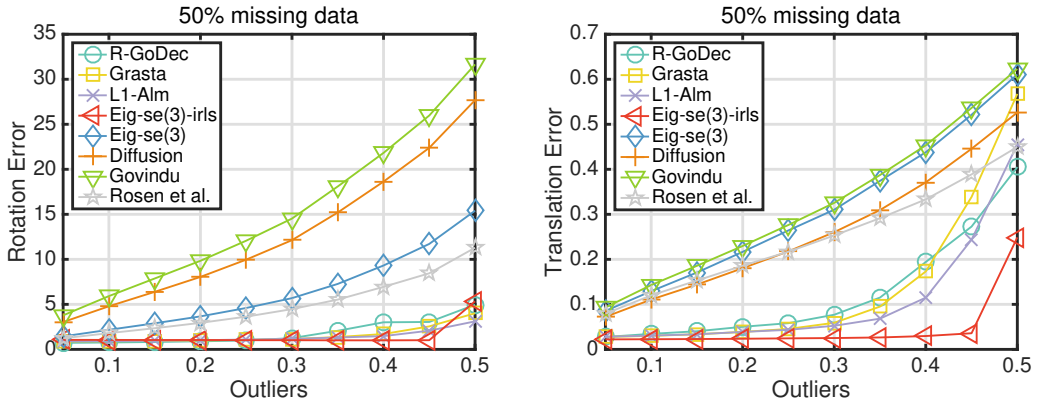


FIGURE 5.12: Mean errors on absolute motions as a function of q, with $p = 0.5$. A fixed level of noise is applied to the inlier relative motions in this experiment.

Missing Data

In this experiment we study how missing data influence the performances of our approaches. We considered $n = 100$ absolute motions sampled as before and we introduced a fixed level of noise on relative motions ($\sigma_T = 0.05, \sigma_R = 5°$). The sparsity parameter $(1 - p)$ ranges from 0.5 to 0.95, which correspond to about 50% and 95% of missing pairs. We considered both the $q = 0$ case (no outliers) and the $q = 0.2$ case. Results are reported in Figure 5.13, which shows the mean errors on absolute motions as a function of the sparsity parameter $(1 - p)$. The errors

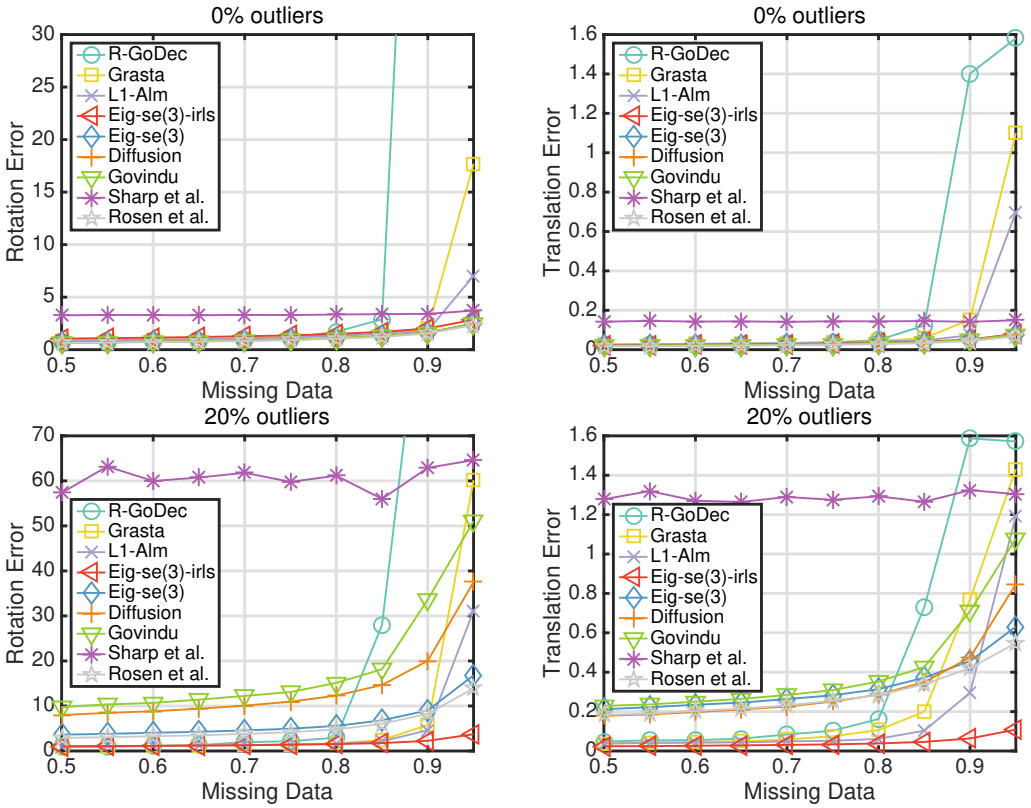


FIGURE 5.13: Mean errors on absolute motions as a function of $(1-p)$, with $q=0$ (top) and $q=0.2$ (bottom). A fixed level of noise is applied to the inlier relative motions in this experiment. In the left sub-figures, the average rotation errors of R-GODEC are approximately 90° for $(1-p)=0.9$ and 120° for $(1-p)=0.95$.

obtained by Sharp et al. remain constant as $(1-p)$ increases, showing no sensitivity to missing data. The same holds for DIFFUSION, the spectral solutions, the methods by Govindu and Rosen et al., if there are no outliers. In the presence of outliers, the errors achieved by such techniques slightly increase as the fraction of missing data becomes higher. As for LRS algorithms, GRASTA and L1-ALM can tolerate up to 90% of missing pairs in the $q=0.2$ case, whereas R-GODEC breaks down with 80% of missing pairs. If there are no outliers ($q=0$), all the LRS methods can tolerate an extra 5% of missing data.

Execution Time

In this experiment we assess the computational efficiency of all the methods in two scenarios. First, we kept the density level of the measurement graph fixed ($p=0.5$) and let n vary between 30 and 600. Then, we kept the number of absolute motions fixed ($n=100$) and let $(1-p)$ vary between 0.05 (about 5% of missing data) and 0.95 (about 95% of missing data). In both cases we introduced a fixed level of noise and outliers on relative motions ($\sigma_T=0.05, \sigma_R=5^\circ$, $q=0.2$). DIFFUSION is implemented in C++ (by the authors), while the remaining algorithms are implemented in MATLAB. Figure 5.14 shows that the method by Sharp et al. is

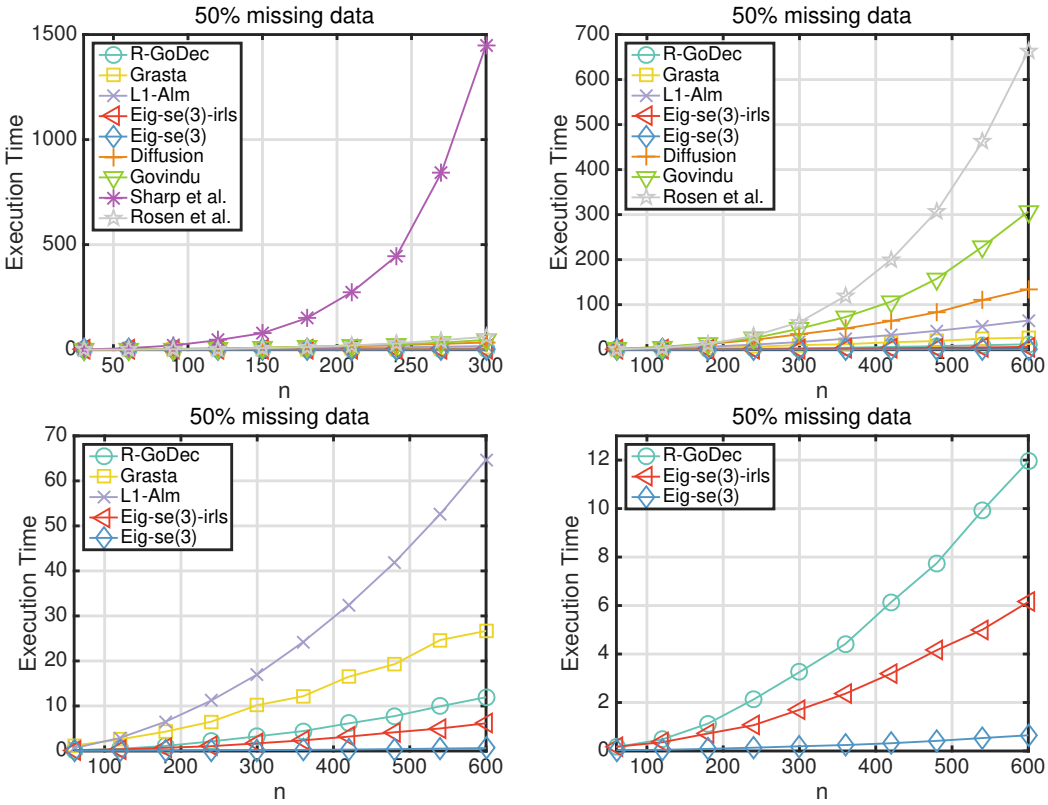


FIGURE 5.14: Execution times [seconds] as a function of n, with $p = 0.5$ and $q = 0.2$. A fixed level of noise is applied to the inlier relative motions in this experiment. The method by Sharp et al. is analyzed only with a maximum of 300 nodes due to computational limitations. The top-right and bottom figures are a magnification of the top-left one.

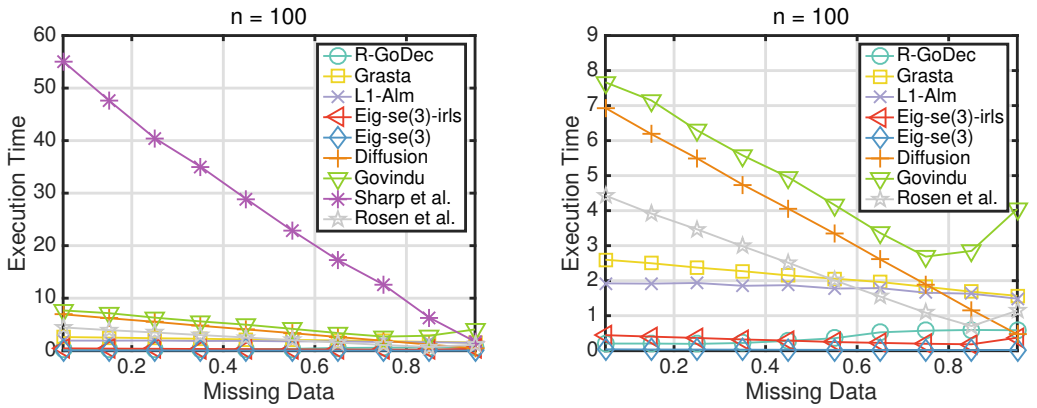


FIGURE 5.15: Execution times [seconds] as a function of $(1 - p)$, with $n = 100$ and $q = 0.2$. A fixed level of noise is applied to the inlier relative motions in this experiment. The right figure is a magnification of the left one.

remarkably slower than the other techniques. R-GODEC, GRASTA and L1-ALM are faster than DIFFUSION and the methods by Govindu and Rosen et al. and slower than the spectral solutions. The R-GODEC algorithm turns out to be the fastest solution among the LRS methods, whereas EIG-SE(3)-IRLS is the fastest among all the robust techniques, computing a solution in 6 seconds for $n = 600$. Figure 5.15

shows that the execution time of matrix decomposition techniques and the spectral solutions do not change significantly when p varies, whereas the other techniques require more time as the measurement graph gets denser.

5.4 Conclusion

In this chapter we reported experiments on synthetic data in order to evaluate the proposed solutions in the context of partial permutation synchronization, rotation synchronization, and rigid-motion synchronization. Results showed that: PARTIAL-SYNCH returns accurate results in the presence of erroneous and missing matches, whereas a previous solution [143] gives accurate results only for total permutations; LRS methods achieve promising results as for robustness to outliers and speed, but they are more affected than the other methods by the sparsity of the graph; EIG-SO(3) and EIG-SE(3) compare favorably with the state of the art in terms of accuracy, they are the fastest solutions among all the analyzed techniques, and, coupled with IRLS, they provide a high resiliency to outliers.

Chapter 6

Real Experiments

In this chapter we evaluate the proposed solutions against state-of-the art algorithms via experiments on real data. In particular, we consider multi-view matching (Section 6.1), multiple point-set registration (Section 6.2) and structure from motion (Section 6.3) as particular applications of the synchronization problem. All the experiments were performed in Matlab on a dual-core MacBook Air with i5 1.3GHz processor and 4Gb RAM.

6.1 Multi-view Matching

In this section we consider the problem of feature matching across multiple images, defined in Section 4.2. The proposed solution (PARTIALSYNCH) was compared to the method in [143] (TOTALSYNCH), as done in Section 5.1. The `Herz-Jesu-P8`, `Entry-P10` and `Fountain-P11` datasets [171] were chosen, which contain 8, 10 and 11 images respectively. A set of features was detected with SIFT [120] in each image using the VLFeat library[1]. Among them a subset was manually selected by looking at the tracks, with the aim of knowing the total number of objects (equal to $d = 30$). Subsequently, correspondences between each image pair (i, j) were established using nearest neighbor and ratio test as in [120] and refined using RANSAC. The resulting relative permutations P_{ij} were given as input to the considered methods.

When evaluating the output, matches were considered correct if they lie within a given distance threshold (set equal to 0.01 times the image diagonal) from the corresponding epipolar lines, computed from the available ground-truth camera matrices. In contrast to the synthetic experiments reported in Section 5.1, the number of matches that should have been returned is not known in real scenarios, hence only the precision could be computed.

TABLE 6.1: Precision [%] achieved by the two methods.

Dataset	PARTIALSYNCH	TOTALSYNCH
`Herz-Jesu-P8`	93.6	50.7
`Entry-P10`	82.0	58.2
`Fountain-P11`	95.4	43.6

[1] `http://www.vlfeat.org/`

The precision achieved by the two methods on these sets is reported in Table 6.1, which confirms the outcome of the experiments on simulated data. For illustration purposes, Figure 6.1 shows the results for some sample images.

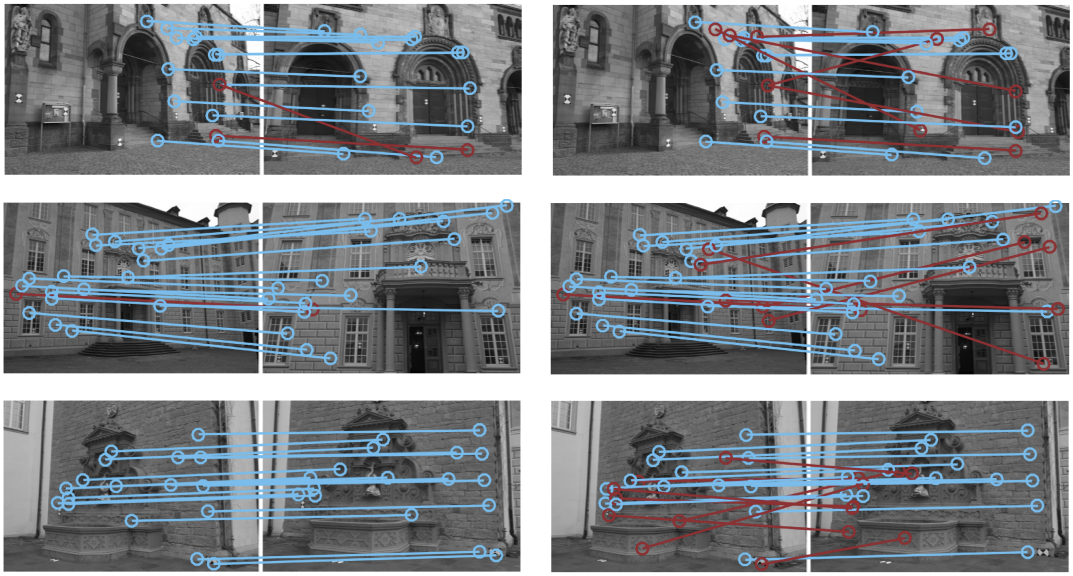


FIGURE 6.1: Matches for a representative image pair from the `Herz-Jesu-P8` dataset (top), from the `Entry-P10` dataset (middle) and from the `Fountain-P11` dataset (bottom), for PARTIALSYNCH (left) and TOTALSYNCH (right). True matches and false matches are shown in light-blue and red, respectively.

6.2 Multiple Point-set Registration

In this section we consider the multiple point-set registration problem, defined in Section 4.3, and we report the outcome of tests on real datasets of range images. As done in Section 5.3, the proposed solutions, namely EIG-SE(3), EIG-SE(3)-IRLS, R-GODEC, GRASTA and L1-ALM, were compared to the methods developed by Sharp et al. [163], Govindu [79], Torsello et al. [177] (DIFFUSION), and Rosen et al. [150].

Relative motion estimates were produced thanks to the MATLAB implementation of ICP (`pcregrigid`). The ICP algorithm has two outputs: an element of SE(3) that transforms one point set to the other; and a registration error computed after applying the transformation. The measurement graph was defined by discarding all the pairs with registration error higher than a threshold. This produced a redundant set of relative motions which were compensated by solving a rigid-motion synchronization problem, returning the transformations that align the original point sets. These estimates could have been improved by alternating rigid-motion synchronization and computing relative motions, as suggested in [177] and [82]. However, such a refinement was not applied in these experiments, i.e. we performed rigid-motion synchronization only once.

Experimentally we observed that LRS methods perform better when translation components have values comparable to rotations, namely in the range $[-1,1]$. For this reason, before performing rigid-motion synchronization, we divided all the relative translations by the maximum of the translations norm (and eventually multiplied the absolute translations by such a scale). This normalization also improves the results of the other algorithms.

TABLE 6.2: Mean errors (rotations in degrees, translations in millimetres) on absolute motions for the Stanford and AIM@SHAPE-VISIONAIR repositories. The number of point sets and the percentage of missing pairs are also reported.

			R-GODEC		GRASTA		L1-ALM		Govindu		DIFFUSION		Sharp et al.		EIG-SE(3)		EIG-SE(3)-IRLS		Rosen et al.	
Dataset	n	% miss.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.	rot.	tra.
Bunny	10	0	0.82	2.9	0.84	1.9	0.78	1.6	1.07	3.7	1.07	3.7	1.07	4.5	1.07	3.7	0.73	1.5	1.07	3.7
Buddha	15	0	0.85	0.3	0.79	0.4	0.94	0.4	1.28	0.4	1.28	0.4	2.22	0.6	1.27	0.4	0.78	0.3	1.27	0.4
Dragon	15	0	0.79	0.3	0.91	0.4	0.77	0.4	1.52	0.4	1.52	0.4	1.45	0.6	1.51	0.4	0.76	0.3	1.52	0.4
Sheep	20	22	0.75	0.8	0.59	1.1	0.34	0.8	5.58	4.2	5.36	4.1	3.31	4.6	4.87	3.8	0.33	0.6	5.04	3.2
Kitten	24	17	1.01	0.8	0.90	0.4	0.91	0.4	1.88	1.4	1.97	1.4	5.71	2.6	1.84	1.4	0.90	0.6	1.85	1.4
Frog	24	23	0.44	1.1	0.26	0.5	0.26	0.4	0.92	1.3	0.92	1.3	2.56	1.7	0.90	1.3	0.28	0.5	0.90	1.3

TABLE 6.3: Execution times (seconds) of rigid-motion synchronization. The number of point sets and the percentage of missing pairs are also reported.

Dataset	n	% miss.	R-GODEC	GRASTA	L1-ALM	Govindu	DIFFUSION	Sharp et al.	EIG-SE(3)	EIG-SE(3)-IRLS	Rosen et al.
Bunny	10	0	0.02	0.17	0.08	0.04	0.09	0.30	0.04	0.34	0.15
Buddha	15	0	0.03	0.24	0.08	0.09	0.20	1.02	0.02	0.12	0.16
Dragon	15	0	0.03	0.26	0.06	0.08	0.18	0.94	0.02	0.12	0.20
Sheep	20	22	0.04	0.32	0.08	0.18	0.29	1.21	0.03	0.11	0.29
Kitten	24	17	0.03	0.44	0.11	0.20	0.42	2.15	0.05	0.13	0.30
Frog	24	23	0.04	0.41	0.10	0.15	0.39	2.44	0.05	0.17	0.27
Gargoyle	27	40	0.05	0.47	0.15	0.13	0.35	1.94	0.03	0.15	0.24
Capital	100	71	0.67	1.62	1.66	0.98	2.53	25.27	0.09	0.38	0.93

Benchmark datasets

In this experiment we considered two benchmark repositories, which provide ground-truth absolute motions in addition to the range images. From the Stanford 3D Scanning Repository[2] we used the `Bunny`, `Happy Buddha` (standing) and `Dragon` (standing) datasets, which contain 10, 15 and 15 point sets, respectively. From the AIM@SHAPE-VISIONAIR Shape Repository[3] we used the `Sheep`, `Kitten` and `Frog` datasets, which contain 20, 24 and 24 point sets, respectively. As for the initialization of the ICP algorithm, we perturbed the available ground-truth motions by a rotation with random axis and angle uniformly distributed over $[0, 2^\circ]$, similarly to the experiments carried out in [82].

Since ground-truth motions are available for these datasets, we evaluated quantitatively the results by reporting the mean errors in Table 6.2. The errors obtained by R-GODEC, GRASTA, L1-ALM and EIG-SE(3)-IRLS are always lower than the other techniques, highlighting the benefit of robustness, and the worst errors are those by

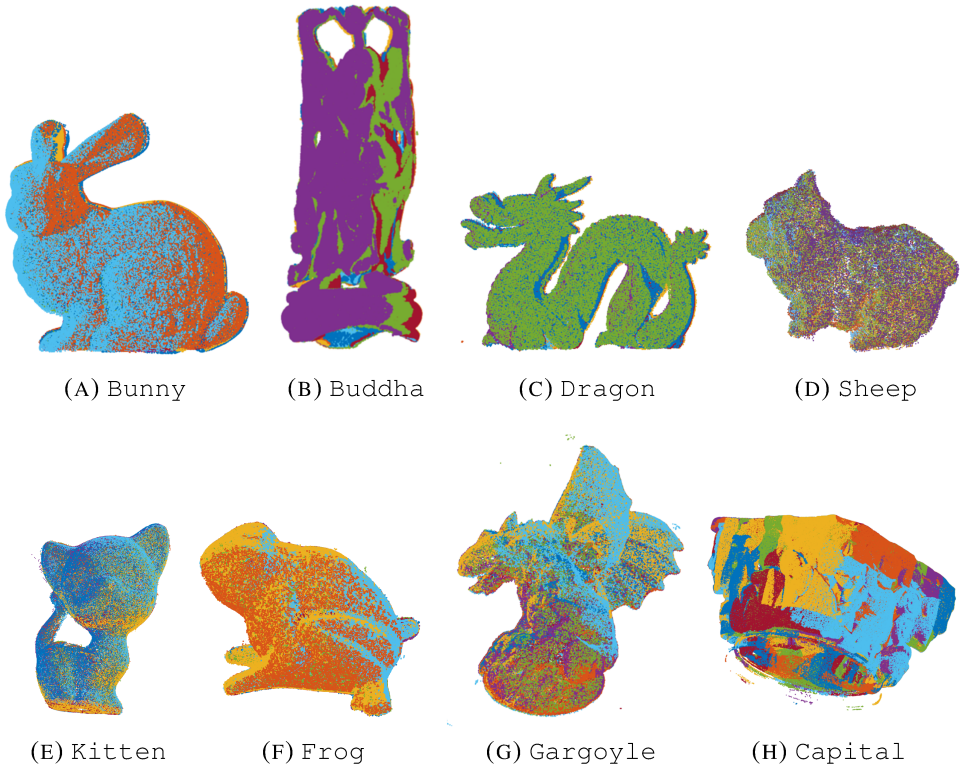
[2] `http://graphics.stanford.edu/data/3Dscanrep/`

[3] `http://visionair.ge.imati.cnr.it/ontologies/shapes/`

TABLE 6.4: Cross-sections of registered point-sets.

Sharp et al.. We also evaluated qualitatively the results in terms of cross-sections in Table 6.4, as it is customary in the registration literature. LRS methods and EIG-SE(3)-IRLS produce accurate cross-sections, which are better than the remaining techniques. The misalignment produced by Sharp et al. is particularly evident in the Kitten dataset. For visualization purposes, we report in Figure 6.2 the 3D models obtained by aligning the original point clouds with L1-ALM.

Execution times are reported in Table 6.3, which refer to the rigid-motion synchronization step, i.e. computing absolute motions from relative motions, and they do not include the time for computing relative motions, which is the same for all the techniques. Table 6.3 shows that the method by Sharp et al. is the slowest one. As for the other techniques, differences in execution times are meaningless for such relatively small datasets.



(A) Bunny (B) Buddha (C) Dragon (D) Sheep

(E) Kitten (F) Frog (G) Gargoyle (H) Capital

FIGURE 6.2: 3D models obtained with L1-ALM. Different point sets are colour coded.

Large-scale datasets

In another experiment we considered two datasets, named `Gargoyle` and `Capital`, which contain 27 and 100 point sets respectively. Since there is no information about the scans, we simply initialized the ICP algorithm with identity matrices. This arbitrary initialization may produce some outliers among relative motions.

Execution times are reported in Table 6.3. R-GODEC is slower than EIG-SE(3) but faster than the other solutions, the method by Sharp et al. is the slowest technique,

while GRASTA and L1-ALM are faster than DIFFUSION, comparable to Rosen et al. and slower than EIG-SE(3)-IRLS and Govindu. The different registration techniques can be appraised qualitatively from the cross-sections of output 3D models reported Table 6.4. The cross-sections obtained by our approach are crisper than the remaining methods, proving the effectiveness of LRS decomposition and EIG-SE(3)-IRLS in handling measurement errors in the context of multiple 3D point-set registration. In particular, the best visual accuracy is achieved by L1-ALM, GRASTA and EIG-SE(3)-IRLS, while R-GODEC get slightly worse results, yet better than the remaining methods. There is no significant difference between the cross-sections obtained by DIFFUSION, EIG-SE(3), Govindu and Rosen et al., while the misalignment produced by Sharp et al. is evident, especially for the Gargoyle dataset. Figure 6.2 shows the 3D models produced by L1-ALM with different colours for each point cloud. In summary, these experiments with real data confirms the conclusions drawn from the simulations.

6.3 Structure from Motion

In this section we address the motion stage of the structure from motion problem, defined in Section 4.4, and we report the outcome of tests on real collections of images, evaluating separately rotations and translations. We considered both the benchmark set up in [171] and the irregular large-scale image collection assembled in [195].

The EPFL benchmark [171] contain from 8 to 30 images and it provides ground-truth absolute motions, which were used to evaluate the performances of the analyzed methods. We followed a common structure-from-motion pipeline to obtain estimates of relative motions. First, reliable matching points across the input images were computed by extracting and matching SIFT features [120]. Then, for each image pair, the essential matrix was computed in a RANSAC scheme, and it was factorized via SVD to obtain the relative rotation and translation direction of the pair. The epipolar graph was built with an edge linking two cameras for which a sufficient number of inlier correspondences was found, and relative motions were improved by applying bundle adjustment to pairs of cameras.

The datasets from [195] contain from 300 to about 900 images and they provide estimates of relative rotations and translation directions, which were given as input to all the analyzed methods. These input motions are very noisy and the associated graphs are highly incomplete in some cases, thus recovering camera motion is challenging. Since ground-truth absolute motions are not available, we used the output of BUNDLER [169] as reference solution, as done in [195].

6.3.1 Rotation Synchronization

As done in Section 5.2, the proposed solutions, namely R-GODEC, GRASTA and L1-ALM, were compared to the spectral solution (EIG) [4] and its robust variation (EIG-IRLS), the semidefinite relaxation (SDP) [4], the rank relaxation (OPTSPACE) [105], the Weiszfeld algorithm [85], the L1-IRLS algorithm [44], and the LUD algorithm [187]. All the methods used the same tuning as in the simulated experiments.

TABLE 6.5: Mean angular errors [degrees] and execution times [seconds] for several algorithms on the datasets from [171]. The number of images and the percentage of missing pairs are also reported.

			GRASTA		R-GODEC		L1-ALM		L1-IRLS		EIG-IRLS		EIG		OPTSPACE		SDP		Weiszfeld		LUD	
Dataset	n	% miss	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time
Castle-P30	30	55	0.55	0.35	1.33	0.04	0.38	0.08	0.65	0.04	0.33	0.32	2.03	0.01	2.06	0.06	2.22	0.40	0.89	0.87	0.56	10.13
Herz-Jesu-P25	25	43	0.68	0.35	0.10	0.02	0.09	0.06	0.06	0.03	0.06	0.11	0.13	0.01	0.14	0.05	0.14	0.30	0.06	0.93	0.07	0.97
Castle-P19	19	58	1.10	0.23	1.80	0.03	0.80	0.04	1.03	0.04	1.34	0.30	2.46	0.01	2.49	0.03	2.52	0.20	1.67	0.37	1.21	1.21
Fountain-P11	11	18	0.04	0.14	0.03	0.02	0.04	0.03	0.03	0.01	0.03	0.15	0.03	0.01	0.03	0.02	0.03	0.21	0.03	0.03	0.03	0.25
Entry-P10	10	2	0.07	0.13	0.43	0.01	0.04	0.03	0.30	0.04	0.05	0.11	0.52	0.01	0.52	0.01	0.52	0.16	0.25	0.10	0.07	1.14
Herz-Jesu-P8	8	18	0.05	0.09	0.05	0.02	0.03	0.06	0.04	0.02	0.04	0.18	0.05	0.01	0.05	0.02	0.05	0.16	0.05	0.02	0.06	0.15

TABLE 6.6: Median angular errors [degrees] and execution times [seconds] for several algorithms on the datasets from [195]. The number of images and the percentage of missing pairs are also reported.

			GRASTA		R-GODEC		L1-ALM		L1-IRLS		EIG-IRLS		EIG		OPTSPACE		SDP		Weiszfeld		LUD	
Dataset	n	% miss	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time	err	time
Vienna Cathedral	918	75	2.08	20	3.11	3.3	1.83	52	1.37	64	1.60	37	5.96	2.2	5.43	5	6.15	2963	3.91	304	-	-
Alamo	627	50	1.25	14	1.47	4.4	1.19	24	1.09	40	1.18	28	3.16	1.2	2.92	1.8	3.21	717	2.11	265	-	-
Notre Dame	553	32	0.84	14	1.04	1.4	0.75	19	0.65	34	0.73	22	3.44	1.1	3.03	1.6	3.65	424	1.87	270	-	-
Tower of London	508	81	2.77	6.2	3.36	0.7	2.90	13.4	2.62	2.9	2.78	6.4	3.86	0.3	3.72	0.9	3.98	380	3.31	73	-	-
Montreal N. Dame	474	53	0.78	8.9	0.86	0.9	0.62	14	0.57	12	0.59	9.9	2.24	0.6	1.87	1.1	2.29	302	1.15	156	0.69	1038
Yorkminster	458	74	2.04	5.9	2.31	3.0	1.85	11.4	1.69	3.3	1.82	6.5	5.84	0.3	4.97	0.9	5.67	280	3.75	80	1.87	937
Madrid Metropolis	394	69	2.92	5	4.11	0.7	2.43	8.4	1.01	7.3	4.43	4	7.48	0.2	6.73	0.5	7.40	187	5.53	66	4.55	695
NYC Library	376	71	2.26	4.7	3.40	0.3	1.76	7.8	1.33	3.2	1.99	3	5.51	0.2	5.28	0.6	5.58	149	3.68	59	1.95	681
Piazza del Popolo	354	60	1.17	4.9	1.63	0.3	1.05	7.1	0.98	5.7	1.03	9.3	3.34	0.2	3.11	0.5	3.48	118	2.27	76	1.22	598

EPFL Benchmark

Results for the EPFL benchmark [171] are shown in Table 6.5, which reports the mean angular errors and execution times of the analyzed methods. R-GODEC, GRASTA and L1-ALM are able to recover camera rotations accurately in low execution time, achieving better results than non-robust algorithms when contamination of outliers is particularly evident, namely in the Castle sequences which contain repetitive structures. In the remaining datasets all the analyzed methods perform well, obtaining an average angular error less than 1°. Among the LRS methods, the highest accuracy is achieved by L1-ALM. Differences in execution times are meaningless for such relatively small datasets, except in the `Castle-P30` sequence where LUD is remarkably slower than the other techniques.

Large-scale Datasets

Results for the datasets from [195] are shown in Table 6.6, which reports the median angular errors and the execution times of the analyzed algorithms. Note that such errors are obtained *without* applying bundle adjustment, which is the final refinement required in any structure from motion system. Neither EIG, SDP and OPTSPACE nor Weiszfeld and LUD are applicable in practical scenarios, since they do not satisfy the requirements of an efficient robust scheme. The first three achieve the highest errors since they are not robust to outliers, whereas the last two show resilience to outliers (to variable degrees) but they have the highest execution times. The remaining algorithms solve the rotation synchronization problem while ensuring robustness and efficiency at the same time, to different extents. In particular, L1-IRLS achieves the highest accuracy, while R-GODEC and GRASTA achieve an accuracy lower than L1-IRLS, albeit comparable in most datasets, in less time. L1-ALM is more accurate than R-GODEC and GRASTA in most image sequences, at the expense of a higher execution time. If the percentage of missing data is high, the performances of LRS methods degrades, in agreement with the outcome of the simulations and the literature on matrix completion. As for EIG-IRLS, results are comparable to L1-IRLS in most datasets both in terms of accuracy and execution time. R-GODEC and GRASTA turn out to be the fastest solutions among all the robust ones.

6.3.2 Translation Recovery

We applied EIG-SE(3)-IRLS and the Group Synchronization Pipeline (GSP)[4] – described in Section 4.4.3 – to the problem of recovering translations in structure from motion. Recall that, owing to the depth-speed ambiguity, the magnitude of relative translations is undefined. In the GSP case, such scales are computed after rotation synchronization, via a divide-et-impera approach that exploits a synchronization instance.

As concerns EIG-SE(3)-IRLS, we exploited three different procedures to compute the epipolar scales. A straightforward approach (suggested in [79]) consists in iteratively updating these scales, i.e. during each iteration the scale of the translation of M_{ij} is set equal to that of $M_i M_j^{-1}$, where M_i and M_j are the current estimates of camera motions. The starting scales are all equal to 1 and the procedure is iterated until convergence. In our implementation this is combined with IRLS in the same loop: in one step we update the IRLS weights and in the next step we update the epipolar scales.

A different approach is described in Section 4.4.4, where we developed a two-stage method for computing the epipolar scales based on the knowledge of relative rotations and translation directions: first, a cycle basis for the epipolar graph is extracted; then all the scales are recovered simultaneously by solving a homogeneous linear system. In our simulations we used a fundamental cycle basis (FCB) in the

[4]The code is available at `http://www.diegm.uniud.it/fusiello/demo/gmf/`

first step. In this way all the unknown norms are computed *before* performing rigid-motion synchronization.

We also provide results obtained by using ground-truth scales (if available), in addition to the approaches mentioned above.

EPFL benchmark

In this experiment we compared EIG-SE(3)-IRLS and GSP to the SDR method [141], whose code is available online, which has an original translation stage and uses EIG to compute rotations. We considered three versions of EIG-SE(3)-IRLS, which differ for the technique chosen to recover the epipolar scales, namely using ground-truth scales (GT), updating the scales iteratively (Iter), and computing the scales through the method in Section 4.4.4 (FCB). As concerns GSP, note that there is no need to divide scale recovery into smaller subproblems, since the EPFL benchmark is made of small-scale datasets. Thus the divide-et-impera technique described in Section 4.4.3 is not used here and GSP reduces to the pipeline sketched in Figure 4.10: first, a rotation synchronization is solved to obtain the angular attitudes of the cameras; then, the epipolar scales are computed as explained in Section 3.2.2; finally, a translation synchronization is performed to recover camera positions. We also included in the comparison the pipeline obtained by combining the R-GoDec algorithm with the bearing-based localization technique by Brand et al. [33]. All the methods were given the same relative motions as input.

TABLE 6.7: Mean errors [meters] on camera translations and execution times [seconds] of rigid-motion synchronization for the datasets from [171]. The number of images and the percentage of missing pairs are also reported.

			EIG-SE(3)-GT		EIG-SE(3)-Iter		EIG-SE(3)-FCB		GSP		SDR		R-GoDec+Brand	
Dataset	n	% miss	err	time	err	time	err	time	err	time	err	time	err	time
CastleP30	30	55	0.150	0.69	2.308	0.65	1.045	0.75	0.581	0.09	1.393	0.70	1.123	0.05
HerzJesuP25	25	43	0.010	0.14	0.931	0.58	0.354	0.18	0.011	0.09	0.065	0.61	0.038	0.05
CastleP19	19	58	0.218	0.41	3.747	0.52	0.853	0.49	1.888	0.08	1.769	0.32	1.493	0.04
FountainP11	11	18	0.002	0.18	0.257	0.42	0.026	0.18	0.007	0.07	0.004	0.23	0.006	0.03
EntryP10	10	2	0.008	0.10	0.307	0.34	0.374	0.12	0.009	0.07	0.203	0.22	0.433	0.03
HerzJesuP8	8	18	0.004	0.24	0.512	0.32	0.014	0.26	0.005	0.07	0.007	0.27	0.009	0.04

Results for the EPFL Benchmark [171] are reported in Table 6.7, which shows the execution times of rigid-motion synchronization and the mean errors on camera translations. The lowest errors are achieved by EIG-SE(3)-GT, confirming the effectiveness of our spectral method for averaging relative motions, when the latter are fully specified. Without ground-truth scales, good estimates of camera translations are still obtained, and precision increases by using FCB rather than the iterative approach. The precision achieved by GSP is comparable to EIG-SE(3)-GT in most datasets. All the methods are able to recover camera motion efficiently, taking less than 1 seconds for all the sequences.

For visualization purposes, we report in Figure 6.3 the 3D structure obtained with GSP on the `Castle-P30` dataset, which was computed via triangulation and subsequently refined with bundle adjustment.

FIGURE 6.3: Left: sample images from the `Castle-P30` dataset. Right: sparse 3D reconstruction obtained with GSP. The root-mean-squared reprojection error (RMSE) is 0.1626 pixels.

Large-scale datasets

We compared EIG-SE(3)-IRLS and GSP with a recent technique – called 1DSfM [195] – which first removes outlier directions by solving simpler low-dimensional sub-problems, and then compute absolute translations through the Levenberg-Marquardt algorithm. Since our MATLAB implementation of the scale recovery via FCB is too slow for large datasets, we did not compute the scales through FCB in this experiment and used the iterative update instead (EIG-SE(3)-Iter). We also included in the comparison the constrained-least-squares (CLS) technique [181], where rotations and translations are initialized separately and then they are jointly refined through Riemannian gradient descent, the least-unsquared-deviations (LUD) solver [140], where absolute translations are derived from a convex program robust to outlier directions, the ShapeKick approach [76], where the Alternating Direction Method of Multipliers is used to robustly compute camera locations, and the method by Cui et al. [56], which extends the technique in [96] by introducing feature tracks. The semi-definite-relaxation (SDR) method [141] has already been shown to have accuracy comparable with LUD with a much higher computing time, thus we did not consider it in these experiments. Our methods, ShapeKick and 1DSfM use as input the relative motions provided in [195] as they are, whereas the remaining pipelines internally refine the pairwise directions, hence they require points in input as well. In LUD such directions are computed through an IRLS scheme robust to outlier point correspondences, whereas in Cui et al. they are obtained through the method described in [107].

Results are shown in Table 6.8, which reports the median translation errors *before* applying bundle adjustment, the number of reconstructed cameras, and the running times. We used the output of BUNDLER [169] as reference solution, and we computed the optimal isometry between this solution and an estimate with least median

of squares (LMedS), using correspondences between camera centres. The results of 1DSfM are taken from [195], the results of LUD and CLS are taken from [140], the results of ShapeKick are taken from [76] and the results by Cui et al. are taken from [56]. In all these papers rotation errors are not analysed, therefore the comparison concentrates on translation errors. We also reported the percentage of missing pairs, which refers to the largest parallel rigid subgraph, extracted as explained in [103].

TABLE 6.8: Median translation errors (metres) on the datasets from [195] *before* bundle adjustment. Times (in seconds) are net of bundle adjustment. The percentage of missing pairs refers to the largest parallel-rigid component of the input graph.

		GSP			EIG-SE(3)-Iter			1DSfM			CLS			LUD			Cui et al.			ShapeKick		
Dataset	miss %	n	tra.	time	n	tra.	time	n	tra.	time	n	tra.	time	n	tra.	time	n	tra.	time	n	tra.	time
Vienna Cathedral	74	684	2.8	69	836	3.2	87	836	6.6	302	836	8.8	578	836	5.4	787	578	3.5	242	836	1.9	156
Alamo	47	499	0.6	40	577	0.7	67	577	1.1	158	577	1.3	239	577	0.4	385	500	0.6	259	577	0.9	68
Notre Dame	32	530	1.5	27	533	0.5	47	553	10	154	553	0.8	512	553	0.3	707	539	0.3	366	553	0.2	68
Tower of London	80	408	1.6	10	572	5.7	17	572	11	78	572	16	55	572	4.7	88	393	4.4	100	472	2.3	24
Montreal Notre Dame	52	423	0.4	14	450	0.7	22	450	2.5	114	450	1.1	180	450	0.5	271	426	0.8	125	450	0.8	32
Yorkminster	72	386	1.4	10	437	6.4	17	437	3.4	122	437	6.2	62	437	2.7	103	341	3.7	45	–	–	–
Madrid Metropolis	65	268	7.5	7	341	8.1	14	341	9.9	43	341	6.4	46	341	1.6	67	–	–	–	341	6.0	19
NYC Library	68	295	1.1	8	332	2.3	12	332	2.5	76	332	5.0	52	332	2.0	102	288	1.4	42	332	1.4	18
Piazza del Popolo	58	297	1.0	14	328	1.1	17	328	3.1	58	328	3.5	62	328	1.5	88	294	2.6	51	338	3.6	17

The median errors reported in Table 6.8 indicate that our methods qualify among the most accurate solutions. In particular, GSP outperforms 1DSfM and CLS, and it places itself in the same range of LUD, ShapeKick and Cui et al. EIG-SE(3) with iterative scale estimate performs equal or better than 1DSfM in most datasets, and it always provides results in the same range of the best methods.

The number of cameras reconstructed by GSP is in general smaller than the other methods. Indeed, some nodes/edges are discarded either because IRLS detects them as outliers or during the clustering phase, which involves rigid components extraction. As a matter of fact, these dataset are taken "in the wild", so one cannot expect to reconstruct all the cameras. The high accuracy achieved by GSP *without any refinement of the input epipolar geometries* confirms that it correctly removes cameras with outlier measures or low connectivity to the rest of the graph.

Computation times of 1DSfM, LUD, ShapeKick and CLS reported in Table 6.8 are obtained by summing the execution costs of rotation and translation recovery, including intermediate steps which are still part of the rigid-motion synchronization pipeline, namely outlier removal in 1DSfM and robust pairwise direction estimation in LUD and CLS. Computation times of the method by Cui et al. are obtained by subtracting the cost for bundle adjustment refinement from the total execution cost. Please note that the execution times are taken from different papers [195, 140, 56] and were obtained on disparate computers. Albeit not directly comparable to each other, these machines are all more powerful than ours, however. Notwithstanding this, results in Table 6.8 demonstrate that GSP and EIG-SE(3)-Iter are the fastest solutions among all the analyzed techniques, and GSP is the fastest overall.

6.4 Conclusion

In this chapter we reported experiments on real data in order to evaluate the proposed solutions in the context of multi-view matching, multiple point-set registration and structure from motion. Results showed that: PARTIALSYNCH handles both false and missing matches which are ubiquitous in multi-view matching, outperforming TOTALSYNCH [143] which gives accurate results only when the same set of features is present in all the images; LRS methods can be profitably applied to multiple point-set registration and to the rotation stage of structure-from-motion, they are computationally undemanding, and they provide a good trade-off between statistical efficiency and resilience to outliers; EIG-SE(3)-IRLS returns accurate estimates of absolute rotations and translations in low execution time, it provides a high resilience to outliers, and it can be successfully applied to multiple point-set registration and – combined with a method for estimating the unknown translation norms – to structure-from-motion; GSP can be profitably used in a global structure-from-motion pipeline that, with respect to its competitors, delivers results of comparable accuracy in substantially less time, thus providing a fast/high-quality initialization for bundle adjustment algorithms.

Chapter 7

Conclusion

In this thesis we studied in depth the synchronization problem, which lies at the basis of several Computer Vision applications. Among them, we considered multi-view matching, multiple point-set registration and structure from motion in the experimental evaluation. The formalism of synchronization, which requires to find elements of a group given measures of their ratios, permits to address these applications without relying on features or points, since the problem is formulated in frame space, or, more abstractly, in a group.

The synchronization problem can be expressed in a compact matrix form, leading to efficient closed form solutions via spectral decomposition, which can be easily made resilient to outliers via Iteratively Reweighted Least Squares. The importance of this result lies in its universality: it can be applied to any group admitting a matrix representation, as opposed to other state-of-the-art techniques which are based on ad-hoc minimizations of specific cost functions, heavily depending on the chosen group. In this respect, we hope that this thesis will serve as a starting point for more research in the field of synchronization.

We also showed that the synchronization problem, under certain circumstances, can be expressed as a low-rank and sparse matrix decomposition, that naturally includes missing data ad outliers in its formulation, and it benefits from a wealth of available algorithms that can be seamlessly used as alternatives. Accordingly, this formulation will benefit from future developments in the field of low-rank and sparse decomposition, which is an active research field. Failure cases appear when the percentage of missing data is extremely high, however, it must be said that the goal of synchronization is to exploit redundancy: if the measures are barely sufficient, the problem starts loosing significance.

Whereas for multi-view matching and multiple point-set registration the formulation of the problem in terms of synchronization is definitive, in the structure from motion case it is not so due to the scale indeterminacy in relative displacements between camera pairs. This brought us to the problem of bearing-based localization, which aims at recovering the position of sensors in a network given measures of pairwise directions. In contrast to the synchronization problem, where the solution is unique (up to a group element) if and only if the underlying graph is connected, in bearing-based localization the solution is unique (up to translation and scale) if and only if the graph is parallel rigid. A comprehensive survey on parallel rigidity is

contained in this thesis, which also includes some novel results and open questions that will be subject of future research.

Appendix A

Kronecker, Hadamard and Khatri-Rao products

This appendix is devoted to illustrate the Kronecker, Hadamard and Khatri-Rao products [184, 130, 118], which are widely used in this thesis.

Let A and B be two real matrices of dimension $m \times r$ and $n \times s$ respectively. The *Kronecker product* of A and B [184], denoted by $A \otimes B$, is defined as

$$A \otimes B = \begin{bmatrix} [A]_{1,1}B & [A]_{1,2}B & \dots & [A]_{1,r}B \\ [A]_{2,1}B & [A]_{2,2}B & \dots & [A]_{2,r}B \\ \dots & & & \dots \\ [A]_{m,1}B & [A]_{m,2}B & \dots & [A]_{m,r}B \end{bmatrix} \tag{A.1}$$

where each $[A]_{i,j}B$ is a block of dimension $n \times s$, thus $A \otimes B$ has dimension $mn \times rs$. The Kronecker product is associative, distributive (with respect to the sum of matrices), but not commutative, and it satisfies the following properties

$$(A \otimes B)^{\mathsf{T}} = A^{\mathsf{T}} \otimes B^{\mathsf{T}} \tag{A.2}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{A.3}$$

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \tag{A.4}$$

$$\mathrm{vec}(AXB) = (B^{\mathsf{T}} \otimes A)\mathrm{vec}(X) \tag{A.5}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization operator which transforms a matrix into a vector by stacking the columns of the matrix one underneath the other.

Let $A = U_A \Sigma_A V_A^{\mathsf{T}}$ and $B = U_B \Sigma_B V_B^{\mathsf{T}}$ be the singular value decompositions of A and B, respectively, then

$$A \otimes B = (U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A \otimes V_B)^{\mathsf{T}} \tag{A.6}$$

which implies

$$\mathrm{rank}(A \otimes B) = \mathrm{rank}(A)\ \mathrm{rank}(B). \tag{A.7}$$

Thus the Kronecker product of two matrices is invertible if and only if both the factors are invertible.

Consider now two real matrices A and B of dimension $m \times r$ and $n \times r$ respectively, and denote the columns of A by $\mathbf{a}_1, \ldots, \mathbf{a}_r$ and those of B by $\mathbf{b}_1, \ldots, \mathbf{b}_r$. The *Khatri-Rao product* of A and B [106, 118], denoted by $A \odot B$, is defined as

$$A \odot B = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_r \otimes \mathbf{b}_r \end{bmatrix} \tag{A.8}$$

where each $\mathbf{a}_i \otimes \mathbf{b}_i$ is a vector of dimension mn, thus $A \odot B$ has dimension $mn \times r$. The Khatri-Rao product is associative, distributive, but not commutative, and it satisfies the following equalities

$$(A \otimes B)(C \odot D) = (AC) \odot (BD) \tag{A.9}$$

$$\operatorname{vec}(A \operatorname{diag}(\mathbf{x}) B) = (B^{\mathsf{T}} \odot A)\mathbf{x} \tag{A.10}$$

where $\operatorname{diag}(\mathbf{x})$ transforms the vector $\mathbf{x} = [x_1 \ \ldots \ x_r]^{\mathsf{T}}$ into a diagonal matrix with elements $x_1, \ldots, x_r$ along the diagonal.

To the best of our knowledge, equalities expressing the rank of $A \odot B$ in terms of the rank of the factors are not present in the literature, in contrast to the case of the Kronecker product. Some inequalities are reported in [106], where it is shown, for instance, that $\operatorname{rank}(A \odot B) \geq \operatorname{rank}(A)$, if all the columns of B corresponding to independent columns of A are non-null.

Let A and B be two real matrices of dimension $m \times r$. The *Hadamard product* (or *entry-wise product*) of A and B [130], denoted by $A \circ B$, has dimension $m \times r$ as well, and it is simply the product of the corresponding elements

$$A \circ B = \begin{bmatrix} [A]_{1,1}[B]_{1,1} & [A]_{1,2}[B]_{1,2} & \cdots & [A]_{1,r}[B]_{1,r} \\ [A]_{2,1}[B]_{2,1} & [A]_{2,2}[B]_{2,2} & \cdots & [A]_{2,r}[B]_{2,r} \\ \cdots & & & \cdots \\ [A]_{m,1}[B]_{m,1} & [A]_{m,2}[B]_{m,2} & \cdots & [A]_{m,r}[B]_{m,r} \end{bmatrix}. \tag{A.11}$$

The Hadamard product is associative, distributive, commutative, and it satisfies the following properties

$$(A \odot B) \circ (C \odot D) = (A \circ C) \odot (B \circ D) \tag{A.12}$$

$$(A \odot B)^{\mathsf{T}}(A \odot B) = (AA^{\mathsf{T}}) \circ (BB^{\mathsf{T}}) \tag{A.13}$$

$$\operatorname{vec}(A \circ B) = \operatorname{diag}(\operatorname{vec}(A)) \operatorname{vec}(B) \tag{A.14}$$

$$\operatorname{diag}(\mathbf{x}) \ A \ \operatorname{diag}(\mathbf{y}) = A \circ (\mathbf{x}\mathbf{y}^{\mathsf{T}}) \tag{A.15}$$

$$\operatorname{rank}(A \circ B) \leq \operatorname{rank}(A) \operatorname{rank}(B). \tag{A.16}$$

Appendix B

Elements of Graph Theory

In this appendix we review some useful concepts from graph theory. A complete treatment of this subject can be found in [43, 26, 100].

A *graph* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a finite set and $\mathcal{E}$ is a family of pairs of elements of $\mathcal{V}$. If the pairs are ordered, then $\mathcal{G}$ is called a *directed graph*, otherwise it is called an *undirected graph*. The elements of $\mathcal{V}$ are called *vertices* or *nodes*, and the elements of $\mathcal{E}$ are called *edges*. We use n and m to denote the number of vertices and edges respectively, namely $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Note that every directed graph can be turned into an undirected graph by ignoring the orientation of the edges, and every undirected graph can be turned into a directed graph by orienting the edges arbitrarily. A *weighted graph* is a graph together with a weight function $\omega : \mathcal{E} \to \mathbb{R}^+$. If the graph is unweighted, we set $\omega : \mathcal{E} \to 1$ and call ω the *uniform weight function*.

An edge $e = (v, w)$ is said to be *incident* to both v and w. If $\mathcal{G}$ is undirected, then v and w are called the *endpoints* of e. If $\mathcal{G}$ is directed, then v and w are called the *tail* and the *head* of e, respectively, and e is said to leave v and enter w. The same edge may occur several times in $\mathcal{E}$. An edge occurring more than once is referred to as a *multiple edge*, and a graph without multiple edges is called *simple*. An edge of the form (v, v) is called a *loop*. In an undirected graph, the *degree* of a vertex v is the number of times that v occurs as an endpoint of an edge. In a directed graph, the *outdegree* and *indegree* of a vertex v are the number of times that v occurs as the tail and head of an edge, respectively.

A *subgraph* $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of $\mathcal{G}$ is a graph with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. If $\mathcal{E}'$ is a subset of $\mathcal{E}$, then $\mathcal{G} \setminus \mathcal{E}'$ denotes the graph obtained by removing all the edges in $\mathcal{E}'$ from $\mathcal{G}$. If $\mathcal{V}'$ is a subset of $\mathcal{V}$, then $\mathcal{G} \setminus \mathcal{V}'$ denotes the graph obtained by removing all the vertices in $\mathcal{V}'$ and their incident edges from $\mathcal{G}$. A *path* from v to w is a subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' = \{v_0 = v, v_1, \ldots, v_k = w\}$ and $\mathcal{E}' = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$. An undirected graph is called *connected* if there exists a path from each vertex to any other, and a directed graph is called connected if the underlying undirected graph is connected. Any maximal connected subgraph $\mathcal{H}$ is called a *connected component*. A graph is a *tree* if it is connected and it has $n - 1$ edges. The disjoint union of trees is called a *forest*. The number of edges in a forest is $n - cc$, where cc denotes the number of connected components in $\mathcal{G}$. A subgraph $\mathcal{G}'$ of a connected graph $\mathcal{G}$ is called a *spanning tree* if it has the same vertices of $\mathcal{G}$

and it is a tree. If $\mathcal{G}$ is not connected, any union of spanning trees for each connected component is called a *spanning forest*.

A connected graph $\mathcal{G}$ is called *biconnected* if it has no articulation points, where a vertex $v \in \mathcal{V}$ is an *articulation point* (or *cut vertex*) if $\mathcal{G} \setminus \{v\}$ is not connected. Any maximal biconnected subgraph is called a *biconnected component*. Equivalently, a biconnected component is a maximal set of edges such that any two edges in the set lie on a common circuit. It can be shown that biconnected components partition the edges of the graph [174], where a single edge is considered biconnected by definition. However, they may share vertices with each other.

A connected graph $\mathcal{G}$ is called *bridgeless* if it has no bridges, where an edge $e \in \mathcal{E}$ is a *bridge* (or *cut edge*) if $\mathcal{G} \setminus \{e\}$ is not connected. It can be shown (e.g. [43]) that if $\mathcal{G}$ is a connected graph on at least 3 vertices and e is a bridge, then e is incident to (at least) one articulation point. In other words, if $\mathcal{G}$ is biconnected and contains at least 3 vertices, then it is bridgeless.

B.1 Cycle Bases

Let $\mathbb{K}$ be a field. A *cycle* in a graph $\mathcal{G}$ is a vector $\mathbf{c} \in \mathbb{K}^m$ such that for any vertex $v \in \mathcal{V}$ it holds

$$\sum_{e \in \delta_+(v)} [\mathbf{c}]_e = \sum_{e \in \delta_-(v)} [\mathbf{c}]_e \tag{B.1}$$

where $\delta_+(v)$ and $\delta_-(v)$ denote the edges leaving and entering v, respectively, and $[\mathbf{c}]_e$ denotes the component of $\mathbf{c}$ indexed by edge e. It is shown in [43] that an edge of a connected graph is a bridge if and only if it does not belong to any cycle. A cycle is *simple* if $[\mathbf{c}]_e \in \{-1, 0, 1\}$ for all $e \in \mathcal{E}$, and a simple cycle is a *circuit* if its support (i.e. the set of edges with $[\mathbf{c}]_e \neq 0$) is connected and for any vertex $v \in \mathcal{V}$ there are at most two edges in the support incident to v. Alternatively, a circuit can be seen a path which starts and ends at the same vertex, in which every vertex appears exactly once (except for the starting vertex which appears exactly twice).

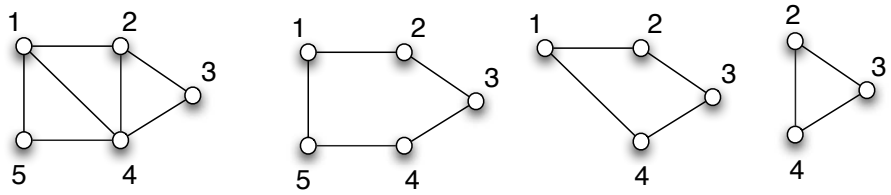


FIGURE B.1: Left: undirected graph with $n = 5$ nodes and $m = 7$ edges. Right: fundamental cycle basis associated to the spanning tree $\mathcal{T} = \{(1,2),(2,3),(3,4),(4,5)\}$.

The set of cycles forms a vector space over $\mathbb{K}$, which is called the *cycle space* of $\mathcal{G}$, and a *cycle basis* is a set of circuits forming a basis of such a space. In general, a cycle basis is not unique. It can be shown [26, 100] that the dimension of the cycle

space is given by the *cyclomatic number*

$$\nu = m - n + cc \tag{B.2}$$

where cc denotes the number of connected components of $\mathcal{G}$. If $\mathcal{G}$ is connected and $\mathcal{T}$ is a spanning tree of $\mathcal{G}$, then adding any edge from $\mathcal{E} \setminus \mathcal{T}$ to $\mathcal{T}$ generates a circuit [100]. The set of such circuits forms a cycle basis, which is referred to as the *fundamental cycle basis*. Figure B.1 reports one example.

Particularly interesting are the cases $\mathbb{K} = \mathbb{Q}$ and $\mathbb{K} = \mathbb{Z}_2$, which correspond to a directed and undirected graph, respectively. If $\mathbb{K} = \mathbb{Q}$, the field of rationals, then the cycle basis is referred to as the *directed cycle basis*. Directed cycles may use arcs in forward ($[\mathbf{c}]_e > 0$) or backward ($[\mathbf{c}]_e < 0$) direction. A directed cycle basis $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\nu\}$ is called a/an

- *integral cycle basis* if each cycle $\mathbf{c}$ of $\mathcal{G}$ can be written as an *integer* linear combination of the circuits in the basis, namely

$$\exists \lambda_i \in \mathbb{Z} : \mathbf{c} = \lambda_1 \mathbf{c}_1 + \lambda_2 \mathbf{c}_2 + \ldots \lambda_\nu \mathbf{c}_\nu; \tag{B.3}$$

- *zero-one cycle basis* (or *totally unimodular cycle basis*) if each cycle $\mathbf{c}$ of $\mathcal{G}$ can be written as a linear combination with coefficients in $\{-1, 0, +1\}$ of the circuits in the basis, namely

$$\exists \lambda_i \in \{-1, 0, +1\} : \mathbf{c} = \lambda_1 \mathbf{c}_1 + \lambda_2 \mathbf{c}_2 + \ldots \lambda_\nu \mathbf{c}_\nu. \tag{B.4}$$

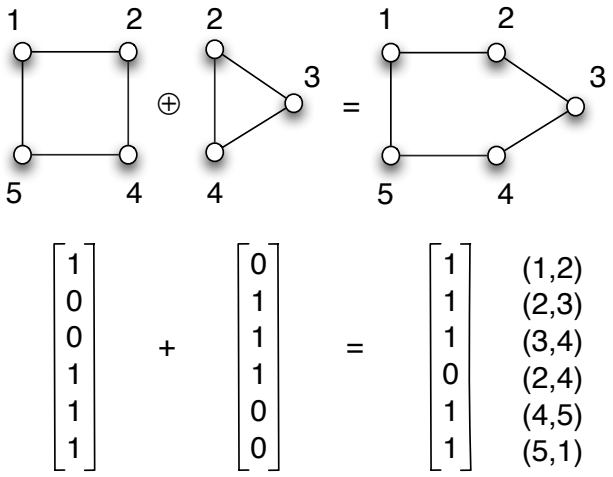


FIGURE B.2: Sum of two cycles.

If $\mathbb{K} = \mathbb{Z}_2 = GF(2)$, the field of two elements, then the cycle basis is referred to as the *undirected cycle basis*. In $\mathbb{Z}_2$ the only non-zero element in the field is

$-1 = +1$, thus a cycle is a vector $\mathbf{c} \in \mathbb{Z}_2^m$ such that for any vertex $v \in \mathcal{V}$ it holds

$$\sum_{e \in \delta(v)} [\mathbf{c}]_e = 0 \tag{B.5}$$

where $\delta(v)$ denotes the set of edges incident to v. Alternatively, an undirected cycle can be viewed as a set of edges, namely it is a subgraph in which every vertex has even degree. The sum of two cycles, denoted by $\oplus$, is a cycle where the common edges vanish, and, more generally, the sum of cycles is the cycle consisting of all the edges that are contained in an odd number in the addends. This concept is illustrated in Figure B.2. It can be shown that an undirected cycle basis can be turned into a directed cycle basis, but the converse is not true [100].

The relationships between the aforementioned classes of cycle bases are illustrated in Figure B.3. The proofs of such inclusions are provided in [100], which also reports counterexamples showing that the inclusions are strict. Note that any directed graph has a basis of each type, since any directed graph has a fundamental cycle basis, and all the other classes generalize fundamental cycle bases.

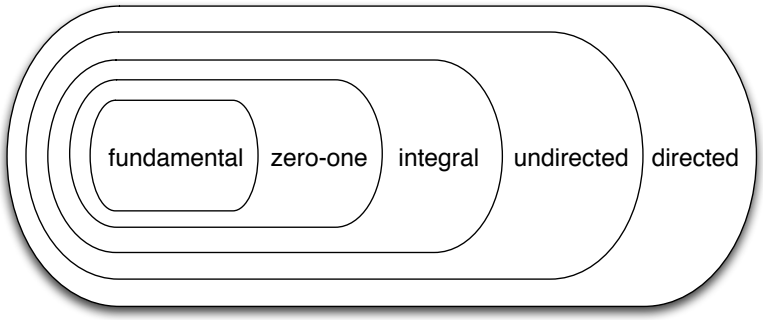


FIGURE B.3: Classification of cycle bases.

B.2 Matrices Associated with Graphs

The *adjacency matrix* A of a graph $\mathcal{G}$ is the $n \times n$ matrix whose elements indicate whether pairs of vertices are adjacent or not, namely

$$[A]_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{B.6}$$

If $\mathcal{G}$ does not contain loops, then A has zero diagonal. Note that the adjacency matrix is symmetric if the graph is undirected.

The *incidence matrix* B of a directed graph $\mathcal{G}$ is the $n \times m$ matrix defined by

$$[B]_{k,e} = \begin{cases} 1 & \text{if } k \text{ is the tail of edge } e, \\ -1 & \text{if } k \text{ is the head of edge } e, \\ 0 & \text{otherwise.} \end{cases} \tag{B.7}$$

The rows of B correspond to vertices and the columns correspond to edges. Note that each column has exactly two non zero entries, which correspond to the endpoints of the edge associated to that column. The *incidence matrix* B of an undirected graph $\mathcal{G}$ is defined considering a particular orientation of the edges. It is shown in [26] that, if $\mathcal{G}$ is connected, then

$$\text{rank}(B) = n - 1. \tag{B.8}$$

The *degree matrix* D of an undirected graph $\mathcal{G}$ is the $n \times n$ diagonal matrix such that $[D]_{i,i}$ contains the degree of node i. Equivalently, it can be defined as

$$D = \text{diag}(A\mathbf{1}_{n\times 1}) \tag{B.9}$$

where $\mathbf{1}_{n\times 1}$ denotes a $n \times 1$ matrix filled by ones, thus $A\mathbf{1}_{n\times 1}$ is the sum of the rows of A. In the case of a directed graph, either the indegree or the outdegree can be used. The *transition matrix* P is defined as

$$P = D^{-1}A. \tag{B.10}$$

The *Laplacian matrix* L of a graph $\mathcal{G}$ is defined as

$$L = D - A. \tag{B.11}$$

It can be checked that, independently of the orientation of the edges, the following equation holds for an undirected graph

$$L = BB^{\mathsf{T}} \tag{B.12}$$

which implies that L is symmetric and positive semidefinite, and, if the graph is connected, $\text{rank}(L) = \text{rank}(B) = n - 1$, hence L is singular. Note that the vector $\mathbf{1}_{n\times 1}$ is in the null-space of L.

The notion of adjacency matrix can be extended to the case of a weighted graph, which translates in letting the entries of A to assume values in $[0, 1]$. Specifically, $[A]_{i,j}$ contains the weight of edge (i, j), and $[A]_{i,j} = 0$ still indicates that $(i, j) \notin \mathcal{E}$. In this case Equations (B.9), (B.10) and (B.11) still make sense, which define the degree matrix, the transition matrix and the Laplacian matrix of a weighted graph, respectively.

The *cycle matrix* C corresponding to a cycle basis of a connected graph $\mathcal{G}$ is the $(m - n + 1) \times m$ matrix having the incidence vectors of the circuits in the basis in its rows. Note that the cycle matrix has columns of zeros in correspondence of bridges

(if they exist). The following equation [26] expresses the relation between the cycle matrix and the incidence matrix

$$CB^{\mathsf{T}} = 0. \tag{B.13}$$

Note that, if the graph is undirected, the matrices A and L are symmetric, thus their eigenvalues are real. The matrix P is not symmetric, but it is similar to the symmetric matrix N defined as

$$N = D^{-1/2}AD^{-1/2} \tag{B.14}$$

since $P = D^{-1/2}ND^{1/2}$. The matrices N and P have the same eigenvalues, so P has real eigenvalues.

Theorem B.1 (Perron-Frobenius [129])**.** *If a $n \times n$ matrix has non-negative entries then it has a non-negative real eigenvalue λ which has maximum absolute value among all the eigenvalues. This eigenvalue has a non-negative real eigenvector. If, in addition, the matrix has no block-triangular decomposition, then λ has multiplicity* 1 *and the corresponding eigenvector is positive.*

As explained in [119], the Perron-Frobenius theorem implies that, if $\mathcal{G}$ is connected, the largest eigenvalue of A has multiplicity 1. Likewise, the largest eigenvalue of the transition matrix is 1 and it ha multiplicity 1. It is easy to check that the eigenvector associated to such eigenvalue is $\mathbf{1}_{n\times 1}$.

Appendix C

Low-rank and Sparse Matrix Decomposition

This appendix provides a comprehensive introduction to "low-rank and sparse" (LRS) matrix decomposition [204]. The idea is that decomposing a data matrix into the sum of terms with specific properties makes the understanding easier as it separates information into simpler pieces. In recent years, decompositions imposing constraints on the rank and sparsity of the addends have become very popular thanks to their profitable application in several fields, such as pattern recognition, machine learning, and signal processing.

Low-rank and sparse decompositions address problems of the general form

$$\mathcal{F}(Z) = \mathcal{F}(L) + S + N \tag{C.1}$$

where Z is a data matrix, $\mathcal{F}$ is a linear operator, L is an unknown low-rank matrix, S is an unknown sparse matrix and N is a diffuse noise. Generally, the sparse term S represents gross errors affecting the measurements (outliers), while the low-rank part represents a meaningful low-dimensional structure contained into the data. The goal is to recover L (and possibly S) under different conditions for S, N and $\mathcal{F}$.

C.1 Robust Principal Component Analysis

An example of LRS decomposition is Robust Principal Component Analysis (RPCA) [37]. The goal is to find the lowest-rank matrix L and the sparsest matrix S such that a given data matrix Z can be decomposed as

$$Z = L + S + N \tag{C.2}$$

with N a diffuse noise. This is illustrated in Figure C.1. Observe that such a decomposition is an instance of the general problem (C.1) with $\mathcal{F}$ being the identity operator.

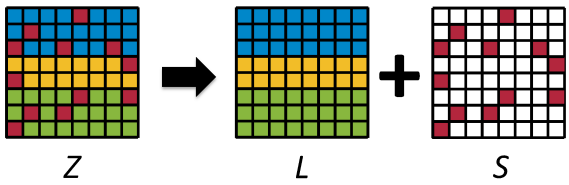


FIGURE C.1: Robust Principal Component Analysis.

A suitable minimization problem for RPCA is

$$\begin{cases} \min\limits_{L,S} \|L\|_* + \lambda \|S\|_1 \\ \text{s.t. } \|Z - L - S\|_F \leq \epsilon \end{cases} \tag{C.3}$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\|\cdot\|_F$ denotes the Frobenius norm, $\|S\|_1$ is the ℓ_1-norm of S (viewed as a vector), and ϵ, λ are given parameters. It is well known from sparse representation theory that minimizing the ℓ_1-norm promotes sparse vectors [68]. Moreover, the nuclear norm is the tightest convex relaxation of the rank function [66], since it is the *sum* of the singular values of a matrix. Thus the solution to problem (C.3) is expected to recover a blind separation between the lowest-rank component and the sparsest errors contained into the data (i.e. the outliers).

Theoretical conditions under which such a solution is stable with respect to a diffuse noise N with high probability are studied in [206] and they depend on some incoherence properties of the data matrix and on the sparsity pattern of S.

Available algorithms for RPCA include, among others, the Accelerated Proximal Gradient (APG) method [206] and extensions of the Augmented Lagrange Multiplier (ALM) method, such as [116] or the ASALM algorithm [173]. These approaches however involve repeated computation of the Singular Value Decomposition (or at least of a partial SVD) of matrices of considerable size which represents the principal bottleneck of current solutions for RPCA.

A faster alternative to RPCA is represented by randomized approximate matrix decomposition [183]. This approach is based on the observation that the low-rank term L of a decomposition of the form (C.2) can be well approximated by random projections onto its column space, thus providing a fast approximation of SVD.

A technique exploiting this paradigm is the GODEC algorithm described in [203]. This method requires to know approximately both the rank r of the low-rank term L and the cardinality (i.e., the number of non-zero entries) k of the sparse term S, and it solves the following minimization problem

$$\begin{cases} \min\limits_{L,S} \|Z - L - S\|_F^2 \\ \text{s.t. } \text{rank}(L) \leq r, \;\; \text{card}(S) \leq k. \end{cases} \tag{C.4}$$

GoDec adopts a *block-coordinate* minimization scheme (also known as *block relaxation* or *alternating optimization*), i.e., it alternatively forces L to the rank-r approximation of $Z - S$, and forces S to the sparse approximation with cardinality k of $Z - L$. The rank-r projection is computed using Bilateral Random Projections (BRP) instead of SVD thus obtaining a speed up in the computation. The updating of S is obtained via entry-wise hard thresholding, keeping the k largest elements of $|Z - L|$ only.

Estimating the cardinality k of the sparse term might be unreliable in practical applications. In order to avoid this parameter, one can consider instead the following minimization problem

$$\begin{cases} \min\limits_{L,S} \dfrac{1}{2} \|Z - L - S\|_F^2 + \lambda \|S\|_1 \\ \text{s.t. } \text{rank}(L) \leq r \end{cases} \tag{C.5}$$

where λ is a regularization parameter which balances the tradeoff between the sparsity of S and the residual error $\|Z - L - S\|_F^2$. In this case, the updating of the sparse part is obtained by minimizing the cost function in (C.5) with respect to S, keeping L constant. Such a problem is known to have an analytical solution, given by the *soft thresholding* or *shrinkage* operator $\Theta_\lambda(\cdot)$ [15] applied to the matrix $Z - L$. This operator is defined as follows

$$\Theta_\lambda(S) = \text{sign}(S) \cdot \max(0, |S| - \lambda) \tag{C.6}$$

where scalar operations are applied element-wise. This method is described in detail in Algorithm 1.

Algorithm 1 GoDec for RPCA

Input: Z, r, ϵ, λ
Output: L, S
Initialize: $L = Z, S = 0$
while $\|Z - L - S\|_F^2 \,/\, \|Z\|_F^2 > \epsilon$ **do**
1. $L \leftarrow$ rank-r approximation of $Z - S$ via BRP
2. $S \leftarrow \Theta_\lambda(Z - L)$

end while

A principled choice of λ, which plays a role similar to an inlier threshold, is derived in [58] in the case of uncorrelated residuals

$$\lambda = \sigma \sqrt{2 \log(m)} \tag{C.7}$$

where m is the number of observations and σ is an estimate of the noise standard deviation.

C.2 Matrix Completion

Robust Principal Component Analysis assumes that the data matrix Z is fully available. However, in practical scenarios, one has to face the problem of missing data. Matrix Completion (MC) is the most natural tool to manage matrices containing unspecified entries [38, 39].

A *partial* matrix is a matrix whose entries are specified on a subset of index pairs and unspecified elsewhere; a *completion* of a partial matrix consists in assigning values to the unspecified entries. Matrix completion problems deal with partial matrices which satisfy some prescribed properties, notably low-rank or positive definiteness. We are concerned here with the low-rank problem, illustrated in Figure C.2, which can be cast as an instance of the general decomposition (C.1) with a specific choice of $\mathcal{F}$ and $S = 0$, namely

$$\mathcal{P}_\Omega(Z) = \mathcal{P}_\Omega(L) + N \tag{C.8}$$

where N has support in Ω. Here Ω is a $(0,1)$-matrix representing the *pattern* (also known as *sampling set*) of Z, i.e., $\Omega_{ij} = 1$ if Z_{ij} is specified and $\Omega_{ij} = 0$ otherwise, and $\mathcal{P}_\Omega(\cdot)$ is the projector operator onto the subspace of matrices that vanish out of Ω, namely $\mathcal{P}_\Omega(X) = \Omega \circ X$, where $\circ$ is the Hadamard (or entry-wise) product, with the provision that an unspecified value multiplied by 0 gives 0.

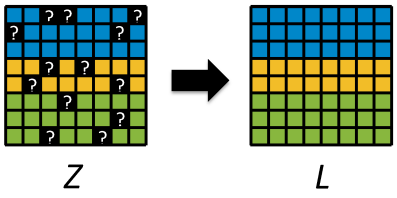


FIGURE C.2: Matrix Completion

The MC problem can be solved through nuclear norm minimization

$$\begin{cases} \min\limits_{L} \|L\|_* \\ \text{s.t. } \|\mathcal{P}_\Omega(Z - L)\|_F \leq \epsilon \end{cases} \tag{C.9}$$

or – if the rank is known a priori – by addressing the following optimization problem

$$\begin{cases} \min\limits_{L} \|\mathcal{P}_\Omega(Z - L)\|_F^2 \\ \text{s.t. } \operatorname{rank}(L) \leq r. \end{cases} \tag{C.10}$$

The minimum sample size required to make matrix completion well-posed is equal to the number of degrees of freedom of the data matrix, namely $(n_1 + n_2 - r)r$, where r is the rank and $n_1 \times n_2$ is the dimension of the matrix. In general, practical MC algorithms do not cope with the minimal case but they require a redundant number of observed entries. For instance, nuclear norm minimization recovers the full low-rank matrix L with high probability – under suitable assumptions – if the

number of observed entries is of the order $O(\bar{n}r\log^6(\bar{n}))$, where $\bar{n} = \max\{n_1, n_2\}$ [39]. Similarly to RPCA, such theoretical conditions depend on some incoherence properties of the data matrix and on the randomness of Ω. With reference to Problem (C.10), the authors of [105] improved the bound on the cardinality of Ω to $O(\bar{n}r\log(\bar{n}))$, with the extra condition that the data matrix has bounded condition number.

Conventional solvers for MC include convex solvers such as ALM [116], SVT [36] and FPCA [122], and subspace identification solvers such as OPTSPACE [105] and ADMiRA [111].

Specifically, in the subspace identification problem the goal is to identify the column space of the unknown low-rank term L. Clearly, any matrix L of rank up to r admits a factorization of the form $L = XY^T$ where X and Y^T are of r columns and r rows respectively. Thus an alternative minimization for the MC problem is

$$\begin{cases} \min\limits_{L,X,Y} \left\| XY^T - L \right\|_F^2 \\ \text{s.t. } \mathcal{P}_\Omega(Z) = \mathcal{P}_\Omega(L). \end{cases} \tag{C.11}$$

In particular, OPTSPACE solves a normalized version of the previous problem, with X, Y belonging to the Grassmannian manifold, namely the set of all r-dimensional subspaces of a Euclidean space, via gradient descent.

The MC problem can also be solved by modifying the GODEC Algorithm, as explained in [203]. The minimization problem (C.10) is reformulated by introducing a sparse term S which approximates $-\mathcal{P}_\mho(L)$, where $\mho$ represents the complementary of Ω, resulting in the following problem

$$\begin{cases} \min\limits_{L,S} \|\mathcal{P}_\Omega(Z) - L - S\|_F^2 \\ \text{s.t. } \text{rank}(L) \leq r,\ \text{supp}(S) = \mho \end{cases} \tag{C.12}$$

where $\text{supp}(S)$ denotes the support of S, i.e., the $(0,1)$-matrix with ij-th entry equal to 1 if $S_{ij} \neq 0$, and equal to 0 otherwise. The associated decomposition problem is

$$\mathcal{P}_\Omega(Z) = L + S + N \tag{C.13}$$

which is equivalent to (C.8) but it does not involve the projection operator $\mathcal{P}_\Omega$ in the right side, thanks to the introduction of the auxiliary variable S. Note that here S does not represent the outliers, but the recovery of missing entries. In the GODEC algorithm for MC, the updating of the sparse term is obtained by assigning $\mathcal{P}_\mho(Z - L) = -\mathcal{P}_\mho(L)$ to S. The method is summarized in Algorithm 2.

Algorithm 2 GoDec for MC

Input: Z, Ω, r, ϵ
Output: L, S
Initialize: $L = Z, S = 0$
while $\|\mathcal{P}_\Omega(Z) - L - S\|_F^2 / \|\mathcal{P}_\Omega(Z)\|_F^2 > \epsilon$ **do**
1. $L \leftarrow$ rank-r approximation of $\mathcal{P}_\Omega(Z) - S$ via BRP
2. $S \leftarrow -\mathcal{P}_\mathcal{O}(L)$

end while

C.3 Robust Matrix Completion

Although being two instances of the same general formulation (C.1), RPCA and MC remain two distinct problems. On one hand, RPCA handles the presence of outlier measurements but it does not deal with missing data, on the other hand MC techniques can fill missing entries, but they are not robust to outliers. Addressing these issues simultaneously is equivalent to solving the following decomposition problem

$$\mathcal{P}_\Omega(Z) = \mathcal{P}_\Omega(L) + S + N \tag{C.14}$$

which aims at recovering the low-rank matrix L starting from an incomplete subset of its entries which are corrupted by both noise and outliers, where S and N have support in Ω. This problem, which is illustrated in Figure C.3, is also known as *robust matrix completion*.

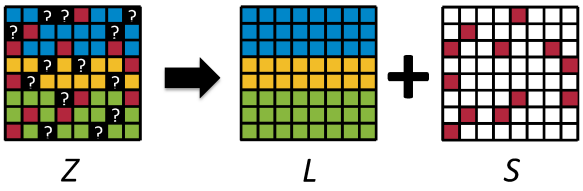


FIGURE C.3: Robust Matrix Completion.

This problem is numerically challenging and poorly studied from a theoretical point of view, as confirmed by the analysis in [204]. A seminal work is presented in [189], where the authors combine a greedy pursuit for updating the sparse term, with an SVD-based approximation for the low-rank term. This method requires to know in advance the cardinality of the sparse term. Other available approaches are [173], which reformulates the problem under the scope of the classical ALM, and [90, 190, 202, 188] which exploit a different formulation in terms of subspace identification in the presence of outliers.

In particular, the GRASTA algorithm presented in [90] minimizes the following cost function

$$\begin{cases} \min\limits_{S,X,Y} \|S\|_1 \\ \text{s.t. } \mathcal{P}_\Omega(Z) = \mathcal{P}_\Omega(XY^T) + S, \end{cases} \tag{C.15}$$

with X belonging to the Grasmannian manifold. GRASTA works on one column of Z at a time, i.e., it considers the following minimization problem

$$\begin{cases} \min\limits_{s,X,y} \|s\|_1 \\ \text{s.t. } \mathcal{P}_\Omega(z) = \mathcal{P}_\Omega(Xy^T) + s, \end{cases} \tag{C.16}$$

where z, s and y are column vectors of Z, S and Y respectively. The Augmented Lagrangian of this constrained minimization problem is

$$\begin{aligned} \mathcal{L}(X, s, y, w) = \|s\|_1 + w^T(\mathcal{P}_\Omega(Xy^T) + s - \mathcal{P}_\Omega(z)) \\ + \frac{\rho}{2}\left\|\mathcal{P}_\Omega(Xy^T) + s - \mathcal{P}_\Omega(z)\right\|^2 \end{aligned} \tag{C.17}$$

where w is the dual vector. GRASTA alternates between estimating X and the triple of vectors (s, y, w). For computing X, GRASTA uses gradient descent on the Grasmannian with (s, y, w) fixed. With X fixed, the triple (s, y, w) is computed using the Alternating Direction Method of Multipliers (ADMM) [32].

The L1-ALM algorithm presented in [202] exploits a similar approach and solves instead

$$\begin{aligned} \min_{X,Y} \quad & \left\|\mathcal{P}_\Omega(Z - XY^T)\right\|_1 + \lambda \left\|Y^T\right\|_* \\ & \text{s.t. } X^T X = I_r. \end{aligned} \tag{C.18}$$

Enforcing X to be column orthogonal shrinks the solution space, while the (convex) nuclear norm regularization term is introduced to improve convergence. The optimization problem is solved via the augmented Lagrange multiplier (ALM) method [21]. At each iteration, the augmented Lagrange function with orthogonal X is minimized using a Gauss-Seidel strategy, then the Lagrange multiplier and the dual parameter are updated.

We introduce here a novel variant of GODEC, dubbed R-GODEC, which manages at the same time both the presence of outliers and unspecified entries in the data matrix Z. More in detail, the sparse term is expressed as the sum of two terms S_1 and S_2 having complementary supports:

- S_1 is a sparse matrix with support on Ω representing outlier measurements;
- S_2 has support on $\mho$ and it is an approximation of $-\mathcal{P}_\mho(L)$, representing completion of missing entries.

This results in the following model

$$\mathcal{P}_\Omega(Z) = L + S_1 + S_2 + N \tag{C.19}$$

which is the natural combination of the RPCA formulation (C.2) with the MC formulation (C.13) associated to the GoDec algorithm. Equation (C.19) reduces to (C.2) over Ω, since S_2 is zero in Ω. On $\mho$ instead, Equation (C.19) turns to $L + S_2 + N = 0$, since both S_1 and Z are zero in $\mho$, and thus S_2 must coincide with $-L$ (up to noise) as in the case of Problem (C.13).

The decomposition Problem (C.19) is translated into the following minimization

$$\begin{cases} \min\limits_{L,S_1,S_2} \frac{1}{2} \|\mathcal{P}_\Omega(Z) - L - S_1 - S_2\|_F^2 + \lambda \|S_1\|_1 \\ \text{s.t. } \text{rank}(L) \leq r, \\ \text{supp}(S_1) \subseteq \Omega, \\ \text{supp}(S_2) = \mho \end{cases} \tag{C.20}$$

which is solved using a block-coordinate scheme that alternates the steps of Algorithm 1 and Algorithm 2. First, the rank-r projection of $\mathcal{P}_\Omega(Z) - S_1 - S_2$ (computed through BRP) is assigned to L. Then, the sparse terms S_1 and S_2 are updated separately. The outlier term S_1 is computed by applying the soft-thresholding operator Θ_λ to the matrix $\mathcal{P}_\Omega(Z - L)$. As for the completion term, $-\mathcal{P}_\mho(L)$ is assigned to S_2, according to the GoDec algorithm for MC. These steps are iterated until convergence or a maximum number of iterations is reached. Our method, called R-GoDec where "R" stands for "robust", is summarized in Algorithm 3.

Algorithm 3 R-GoDec

Input: Z, Ω, r, ϵ, λ
Output: L, S_1, S_2
Initialize: $L = Z$, $S_1 = 0$, $S_2 = 0$
while $\|\mathcal{P}_\Omega(Z) - L - S_1 - S_2\|_F^2 \,/\, \|\mathcal{P}_\Omega(Z)\|_F^2 > \epsilon$ **do**
1. $L \leftarrow$ rank-r approximation of $\mathcal{P}_\Omega(Z) - S_1 - S_2$ via BRP
2. $S_1 \leftarrow \Theta_\lambda(\mathcal{P}_\Omega(Z - L))$
3. $S_2 \leftarrow -\mathcal{P}_\mho(L)$

end while

In some applications the data matrix has a block structure, and this should be reflected by the sparse term which represents the outliers. This is taken into account by modifying Algorithm 3 in order to enforce a block-structure in S_1. Specifically, the ℓ_1-norm in (C.20) is substituted with the mixed $\ell_{2,1}$-norm which promotes group sparsity, resulting in the following problem

$$\begin{cases} \min\limits_{L,S_1,S_2} \frac{1}{2} \|\mathcal{P}_\Omega(Z) - L - S_1 - S_2\|_F^2 + \lambda \|S_1\|_{2,1} \\ \text{s.t. } \text{rank}(L) \leq r, \\ \text{supp}(S_1) \subseteq \Omega, \\ \text{supp}(S_2) = \mho \end{cases} \tag{C.21}$$

where the mixed $\ell_{2,1}$-norm of a matrix S is the sum of the Frobenius norm of each block S_{ij}

$$\|S\|_{2,1} = \sum_{i,j} \|S_{ij}\|_F . \tag{C.22}$$

The minimum of the cost function in (C.21) with respect to S_1 keeping the other variables constant has a closed-form expression, given by the *generalized soft-thresholding* (or shrinkage) operator $\Theta_\lambda^{2,1}$ applied to the matrix $\mathcal{P}_\Omega(Z - L)$ [108]. Such an operator takes a matrix S as input and on each block S_{ij} it computes

$$\Theta_\lambda^{2,1}(S_{ij}) = S \cdot \max(1 - \frac{\lambda}{\|S_{ij}\|_F}, 0) \tag{C.23}$$

where scalar operations are applied element-wise. In this way the selected blocks are the ones with the biggest Frobenius norm. Accordingly, Step 2 of Algorithm 3 is modified as follows

$$S_1 \leftarrow \Theta_\lambda^{2,1}(\mathcal{P}_\Omega(Z - L)). \tag{C.24}$$

Please note that, as in Algorithm 3, the cost function in (C.21) has a unique minimum with respect to S_1. The flexible structure of R-GoDec also allows an extension to the case where the data matrix contains Euclidean distances [152].

Bibliography

[1] K. Aftab, R. Hartley, and J. Trumpf. Generalized Weiszfeld algorithms for l_q optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(37):728 – 745, 2015.

[2] M. Agrawal. A Lie-algebraic approach for consistent pose registration for general euclidean motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1891–1897, 2006.

[3] R. Aragues, L. Carlone, C. Sagues, and G. Calafiore. Distributed centroid estimation from noisy relative measurements. *Systems and Control Letters*, 61(7):773 – 779, 2012.

[4] M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri. Global motion estimation from point matches. *Proceedings of the Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2012.

[5] F. Arrigoni, A. Fusiello, and B. Rossi. On computing the translations norm in the epipolar graph. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 300–308, 2015.

[6] F. Arrigoni, A. Fusiello, and B. Rossi. Camera motion from group synchronization. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 546–555, 2016.

[7] F. Arrigoni, L. Magri, B. Rossi, P. Fragneto, and A. Fusiello. Robust absolute rotation estimation via low-rank and sparse matrix decomposition. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 491–498, 2014.

[8] F. Arrigoni, E. Maset, and A. Fusiello. Synchronization in the symmetric inverse semigroup. In *Proceedings of the International Conference on Image Analysis and Processing*, 2017.

[9] F. Arrigoni, B. Rossi, and A. Fusiello. Global registration of 3D point sets via LRS decomposition. In *Proceedings of the 14th European Conference on Computer Vision*, pages 489–504, 2016.

[10] F. Arrigoni, B. Rossi, and A. Fusiello. Spectral synchronization of multiple views in SE(3). *SIAM Journal on Imaging Sciences*, 9(4):1963 – 1990, 2016.

[11] F. Arrigoni, B. Rossi, F. Malapelle, P. Fragneto, and A. Fusiello. Robust global motion estimation with matrix completion. *ISPRS - International Archives of*

the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-5:63–70, 2014.

[12] J. Aspnes, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, Y. Yang, B. Anderson, and P. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663 – 1678, 2006.

[13] P. Barooah and J. P. Hespanha. Estimation on graphs from relative measurements. *IEEE Control Systems*, 27(4):57 – 74, 2007.

[14] P. Barooah and J. P. Hespanha. Estimation from relative measurements: Electrical analogy and large graphs. *IEEE Transactions on Signal Processing*, 56(6):2181 – 2193, 2008.

[15] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, March 2009.

[16] A. Beinat and F. Crosilla. Generalized procrustes analysis for size and shape 3D object reconstruction. In *Optical 3-D Measurement Techniques*, pages 345–353. Wichmann Verlag, 2001.

[17] C. Belta and V. Kumar. Euclidean metrics for motion generation on SE(3). *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 216(1):47–60, 2002.

[18] R. Benjemaa and F. Schmitt. A solution for the registration of multiple 3D point sets using unit quaternions. In *Proceedings of the European Conference on Computer Vision*, pages 34–50, 1998.

[19] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multiview registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):540–547, May 1996.

[20] F. Bernard, J. Thunberg, P. Gemmar, F. Hertel, A. Husch, and J. Goncalves. A solution for multi-alignment by transformation synchronisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[21] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[22] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[23] B. Bhowmick, S. Patra, A. Chatterjee, V. M. Govindu, and S. Banerjee. Divide and conquer: Efficient large-scale structure from motion using graph partitioning. In *12th Asian Conference on Computer Vision (ACCV 2014)*, 2014.

[24] A. N. Bishop, B. D. O. Anderson, B. Fidan, P. N. Pathirana, and G. Mao. Bearing-only localization using geometrically constrained optimization. *IEEE Transactions on Aerospace and Electronics Systems*, 45(1):308 – 320, 2009.

[25] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks*, 2(2):188–220, 2006.

[26] B. Bollobas. *Modern Graph Theory*. Springer, 1998.

[27] F. Bonarrigo and A. Signoroni. An enhanced 'optimization-on-a-manifold' framework for global registration of 3D range data. *Proceedings of the Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 350 – 357, 2011.

[28] N. Boumal, A. Singer, and P.-A. Absil. Robust estimation of rotations from relative measurements by maximum likelihood. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[29] N. Boumal, A. Singer, P. A. Absil, and V. D. Blondel. Cramer-Rao bounds for synchronization of rotations. *Information and Inference: A Journal of the IMA*, 3(1):1 – 39, 2014.

[30] G. Bourmaud. Online variational Bayesian motion averaging. In *Proceedings of the European Conference on Computer Vision*, pages 126 – 142, 2016.

[31] G. Bourmaud, R. Megret, A. Giremus, and Y. Berthoumieu. Global motion estimation from relative measurements in the presence of outliers. In *Proceedings of the Asian Conference on Computer Vision*, 2014.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, Jan. 2011.

[33] M. Brand, M. Antone, and S. Teller. Spectral solution of large-scale extrinsic camera calibration as a graph embedding problem. In *Proceedings of the European Conference on Computer Vision*, 2004.

[34] M. Brown and D. G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, June 2005.

[35] N. Brown, J. M. Bull, and I. Bethune. Solving large sparse linear systems using asynchronous multisplitting. Technical report, EPCC, The University of Edinburgh, 2013.

[36] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[37] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11:1–11:37, June 2011.

[38] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[39] E. J. Candès and T. Tao. The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[40] J. Cardoso and F. S. Leite. On computing the logarithm in the special Euclidean group of motions in $\mathbb{R}^n$. preprint, Departamento de Matematica, 99-01, Universidade de Coimbra, 1999.

[41] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert. Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015.

[42] U. Castellani, A. Fusiello, and V. Murino. Registration of multiple acoustic range views for underwater scene reconstruction. *Computer Vision and Image Understanding*, 87(1):78–89, 2002.

[43] G. Chartrand. *Introductory Graph Theory*. Dover, New York, 1985.

[44] A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. In *Proceedings of the International Conference on Computer Vision*, 2013.

[45] A. Chatterjee and V. M. Govindu. Robust relative rotation averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[46] K. N. Chaudhury, Y. Khoo, and A. Singer. Global registration of multiple point clouds using semidefinite programming. *SIAM Journal on Optimization*, 25(1):468 – 501, 2015.

[47] Y. Chen, L. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. In *Proceedings of the International Conference on Machine Learning*, pages 100–108, 2014.

[48] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2724 – 2729, 1991.

[49] M. Chudnovsky, W. H. Cunningham, and J. Geelen. An algorithm for packing non-zero a-paths in group-labelled graphs. *Combinatorica*, 28(2):145–161, 2008.

[50] R. Connelly. Generic global rigidity. *Discrete and Computational Geometry*, 33(4):549 – 563, 2004.

[51] L. Cosmo, A. Albarelli, F. Bergamasco, A. Torsello, E. Rodolà, and D. Cremers. A game-theoretical approach for joint matching of multiple feature throughout unordered images. In *Proceedings of the International Conference on Pattern Recognition*, pages 57–60, 2016.

[52] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3001–3008, 2011.

[53] F. Crosilla and A. Beinat. Use of generalised procrustes analysis for the photogrammetric block adjustment by independent models. *ISPRS Journal of Photogrammetry & Remote Sensing*, 56(3):195–209, 2002.

[54] M. Cucuringu. Synchronization over Z_2 and community detection in signed multiplex networks with constraints. *Journal of Complex Networks*, 3(3):469–506, 2015.

[55] M. Cucuringu, Y. Lipman, and A. Singer. Sensor network localization by eigenvector synchronization over the Euclidean group. *ACM Transactions on Sensor Networks*, 8(3):19:1 – 19:42, 2012.

[56] Z. Cui, N. Jiang, C. Tang, and P. Tan. Linear global translation estimation with feature tracks. In *British Machine Vision Conference*, 2015.

[57] M. Cygan, M. Pilipczuk, and M. Pilipczuk. On group feedback vertex set parameterized by the size of the cutset. In *Graph-Theoretic Concepts in Computer Science - 38th International Workshop*, pages 194–205, 2012.

[58] D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[59] G. Dubbelman and B. Browning. COP-SLAM: Closed-form online pose-chain optimization for visual SLAM. *IEEE Transactions on Robotics*, 31(5):1194 – 1213, 2015.

[60] P. H. Edelman and M. Saks. Group labelings of graphs. *Journal of Graph Theory*, 3(2):135–140, 1979.

[61] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *Eleventh Workshop on Omnidirectional Vision, Camera Networks and Non-classical Camera*, 2011.

[62] T. Eren, W. Whiteley, and P. N. Belhumeur. A theoretical analysis of the conditions for unambiguous node localization in sensor networks. Technical Report CUCS-O32-M, Columbia University, Computer Science Department, 2004.

[63] T. Eren, W. Whiteley, and P. N. Belhumeur. Using angle of arrival (bearing) information in network localization. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4676 – 4681, 2006.

[64] T. Eren, W. Whiteley, A. S. Morse, P. N. Belhumeur, and B. D. O. Anderson. Sensor and network topologies of formations with direction, bearing and angle information between agents. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3064 – 3069, 2003.

[65] S. Fantoni, U. Castellani, and A. Fusiello. Accurate and automatic alignment of range surfaces. In *Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 73 – 80, 2012.

[66] M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

[67] A. Fitzgibbon. Robust registration of 2D and 3D point sets. *Image and Vision Computing*, 21(13-14):1145 – 1153, 2003.

[68] M. Fornasier. *Theoretical Foundations and Numerical Methods for Sparse Recovery*. Radon Series on Computational and Applied Mathematics. De Gruyter, 2010.

[69] C. Fraser. Network orientation models for image-based 3D measurement. *ISPRS Archives*, XXXVI-5/W17, 2005.

[70] J. Fredriksson and C. Olsson. Simultaneous multiple rotation averaging using Lagrangian duality. In *Proceedings of the Asian Conference on Computer Vision*, 2012.

[71] A. Fusiello, U. Castellani, L. Ronchetti, and V. Murino. Model acquisition by registration of multiple acoustic range views. In *Proceedings of the European Conference on Computer Vision*, pages 805–819, 2002.

[72] A. Fusiello and F. Crosilla. Solving bundle block adjustment by generalized anisotropic Procrustes analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 102:209–221, April 2015.

[73] M. Gavish and A. Weiss. Performance analysis of bearing-only target location algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):817 – 828, 1992.

[74] R. Gherardi, M. Farenzena, and A. Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1594 – 1600, 13-18 June 2010.

[75] A. Giridhar and P. Kumar. Distributed clock synchronization over wireless networks: Algorithms and analysis. *Proceedings of the IEEE Conference on Decision and Control*, pages 4915–4920, 2006.

[76] T. Goldstein, P. Hand, C. Lee, V. Voroninski, and S. Soatto. ShapeFit and ShapeKick for robust, scalable structure from motion. In *Proceedings of the European Conference on Computer Vision*, pages 289 – 304, 2016.

[77] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[78] V. M. Govindu. Combining two-view constraints for motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[79] V. M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 684–691, 2004.

[80] V. M. Govindu. Robustness in motion averaging. In *Proceedings of the Asian Conference on Computer Vision*, pages 457–466, 2006.

[81] V. M. Govindu. Motion averaging: a framework for efficient and accurate large-scale camera estimation in 3D vision. Tutorial at CVPR, 2017. `http://www.ee.iisc.ac.in/labs/cvl/cvpr2017/tutorial/`.

[82] V. M. Govindu and A. Pooja. On averaging multiview relations for 3D scan registration. *IEEE Transactions on Image Processing*, 23(3):1289–1302, 2014.

[83] S. Guillemot. FTP algorithms for path-traversal and cycle-traversal problems. *Discrete Optimization*, 8(1):61 – 71, 2011. Parameterized Complexity of Discrete Optimization.

[84] R. Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63:129 – 137, 2002.

[85] R. Hartley, K. Aftab, and J. Trumpf. L1 rotation averaging using the Weiszfeld algorithm. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3041–3048, 2011.

[86] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision*, 2013.

[87] R. I. Hartley. In defence of the 8-point algorithm. In *Proceedings of the International Conference on Computer Vision*, pages 1064–1071. IEEE Computer Society, 1995.

[88] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[89] M. Havlena and K. Schindler. VocMatch: Efficient multiview correspondence for structure from motion. In *Proceedings of the European Conference on Computer Vision*, pages 46 – 60, 2014.

[90] J. He, L. Balzano, and A. Szlam. Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1568 – 1575, 2012.

[91] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977.

[92] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Eurographics Symposium on Geometry Processing*, volume 32, pages 177–186, 2013.

[93] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137:346 – 365, 1997.

[94] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In *IEEE INFOCOM*, 2004.

[95] K. Jia, T.-H. Chan, Z. Zeng, S. Gao, G. Wang, T. Zhang, and Y. Ma. ROML: A robust feature correspondence approach for matching objects in a set of images. *International Journal of Computer Vision*, 117(2):173–197, 2016.

[96] N. Jiang, Z. Cui, and P. Tan. A global linear method for camera pose registration. In *Proceedings of the International Conference on Computer Vision*, 2013.

[97] M. Joglekar, N. Shah, and A. A. Diwan. Balanced group-labeled graphs. *Discrete Mathematics*, 312(9):1542 – 1549, 2012. Recent Trends in Graph Theory and Combinatorics.

[98] R. Karp, J. Elson, D. Estrin, and S. Shenker. Optimal and global time synchronization in sensor nets. Technical report, Center for Embedded Networked Sensing: University of California, Los Angeles, 2003.

[99] B. Katz, M. Gaertler, and D. Wagner. Maximum rigid components as means for direction-based localization in sensor networks. In *Annual Conference on Current Trends in Theory and Practice of Computer Science*, pages 330–341, 2007.

[100] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. Zweig. Cycle bases in graphs: Characterization, algorithms, complexity, and applications. *Computer Scienze Review*, 3(4):199–243, 2009.

[101] K. Kawarabayashia and P. Wollan. Non-zero disjoint cycles in highly connected group labelled graphs. *Journal of Combinatorial Theory, Series B*, 96(2):296 – 301, 2006.

[102] J. Keller. Closest unitary, orthogonal and Hermitian operators to a given operator. *Mathematics Magazine*, 48:192–197, 1975.

[103] R. Kennedy, K. Daniilidis, O. Naroditsky, and C. J. Taylor. Identifying maximal rigid components in bearing-based localization. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 194 – 201, 2012.

[104] R. Kennedy and C. J. Taylor. Network localization from relative bearing measurements. In *International Conference on Intelligent Robots and Systems*, pages 149 – 156, 2014.

[105] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[106] C. G. Khatri and C. R. Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhya: The Indian Journal of Statistics, Series A (1961-2002)*, 30(2):pp. 167–180, 1968.

[107] L. Kneip, M. Chili, and R. Siegwart. Robust real-time visual odometry with a single camera and an IMU. In *British Machine Vision Conference*, 2011.

[108] M. Kowalski and B. Torrésani. Structured sparsity: from mixed norms to structured shrinkage. *Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[109] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian. Optimisation-on-a-manifold for global registration of multiple 3D point sets. *International Journal of Intelligent Systems Technologies and Applications*, 3(3/4):319–340, 2007.

[110] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 2:83 – 97, 1955.

[111] K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Transactions on Information Theory*, 56(9):4402 – 4416, 2010.

[112] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the International Conference on Computer Vision*, pages 1482 – 1489, 2005.

[113] N. Levi and M. Werman. The viewing graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 518 – 522, 2003.

[114] K. Li, Y. Xi, and Y. Zhang. Some novel characterizations of generic rank of structured matrix. In *IEEE American Control Conference*, pages 2510 – 2514, 1998.

[115] W.-Y. D. Lin, M.-M. Cheng, J. Lu, H. Yang, M. N. Do, and P. Torr. Bilateral functions for global motion modeling. In *Proceedings of the European Conference on Computer Vision*, pages 341–356. Springer International Publishing, 2014.

[116] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted Low-Rank matrices. eprint arXiv:1009.5055, 2010.

[117] Y. Lipman, S. Yagev, R. Poranne, D. W. Jacobs, and R. Basri. Feature matching with bounded distortion. *ACM Transactions on Graphics*, 33(3):26:1–26:14, 2014.

[118] S. Liu and G. Trenkler. Hadamard, Khatri-Rao, Kronecker and other matrix products. *International Journal of Information and Systems Sciences*, 4(1):160 – 177, 2008.

[119] L. Lovász. Eigenvalues of graphs. Technical report, 2007.

[120] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[121] J. Ma, W. Qiu, J. Zhao, Y. Ma, A. L. Yuille, and Z. Tu. Robust L_2E estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing*, 63(5):1115 – 1129, 2015.

[122] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[123] J. Maciel and J. P. Costeira. A global solution to sparse correspondence problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):187 – 199, 2003.

[124] F. Malapelle, A. Fusiello, B. Rossi, E. Piccinelli, and P. Fragneto. Uncalibrated dynamic stereo using parallax. In *International Symposium on Image and Signal Processing and Analysis (ISPA)*. IEEE, 2013.

[125] G. Mao, B. Fidan, and B. D. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529 – 2553, 2007.

[126] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[127] E. Maset, F. Arrigoni, and A. Fusiello. Practical and efficient multi-view matching. In *Proceedings of the International Conference on Computer Vision*, 2017.

[128] X. Mateo, X. Orriols, and X. Binefa. Bayesian perspective for the registration of multiple 3D views. *Computer Vision and Image Understanding*, 118:84 – 96, 2014.

[129] C. D. Meyer. *Matrix Analysis and applied linear algebra*. SIAM, 2000.

[130] T. Minka. Old and new matrix algebra useful for statistics. MIT Media Lab note, 2000. http://research.microsoft.com/ minka/papers/matrix/.

[131] M. Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 4(1):1–16, 2002.

[132] P. Molavi and A. Jadbabaie. A topological view of estimation from noisy relative measurements. In *IEEE American Control Conference*, 2011.

[133] F. Mosteller and J. W. Tukey. *Data analysis and regression: a second course in statistics*. Addison-Wesley Series in Behavioral Science: Quantitative Methods, 1977.

[134] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the International Conference on Computer Vision*, pages 3248–3255, 2013.

[135] K. Ni and F. Dellaert. Hypersfm. In *Proceedings of the Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 144–151, 2012.

[136] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *IEEE INFOCOM*, pages 4104 – 4113, 2003.

[137] R. Oliveira, R. Ferreira, and J. P. Costeira. Optimal multi-frame correspondence with assignment tensors. In *Proceedings of the European Conference on Computer Vision*, pages 490–501, 2006.

[138] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Proceedings of the 17th Scandinavian conference on Image analysis (SCIA'11)*, pages 524–535. Springer-Verlag, 2011.

[139] O. Ozyesil. *Camera motion estimation by convex programming*. PhD thesis, Princeton University, 2014.

[140] O. Ozyesil and A. Singer. Robust camera location estimation by convex programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2674 – 2683, 2015.

[141] O. Ozyesil, A. Singer, and R. Basri. Stable camera motion estimation using convex programming. *SIAM Journal on Imaging Sciences*, 8(2):1220 – 1262, 2015.

[142] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer. A survey of structure from motion. *Acta Numerica*, 26:305 – 364, 2017.

[143] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems 26*, pages 1860–1868. 2013.

[144] X. Pennec. Multiple registration and mean rigid shape: Applications to the 3D case. In *16th Leeds Annual Statistical Workshop*, pages 178–185, 1996.

[145] J. R. Peters, D. Borra, B. E. Paden, and F. Bullo. Sensor network localization on the group of three-dimensional displacements. *SIAM Journal on Control and Optimization*, 53(6):3534–3561, 2015.

[146] A. Postnikov and A. Spiridonov. The parallel rigidity index of a graph. Preprint, 2007.

[147] K. Pulli. Multiview registration for large data sets. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 160–168, 1999.

[148] A. C. Raghuramu. Robust multiview registration of 3D surfaces via ℓ_1-norm minimization. In *British Machine Vision Conference*, 2015.

[149] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint probabilistic matching using m-best solutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–145, 2016.

[150] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard. A certifiably correct algorithm for synchronization over the special Euclidean group. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[151] D. M. Rosen, C. DuHadway, and J. J. Leonard. A convex relaxation for approximate global optimization in simultaneous localization and mapping. In

Proceedings of the IEEE International Conference on Robotics and Automation, pages 5822 – 5829, 2015.

[152] B. Rossi, M. Patanè, P. Fragneto, and A. Fusiello. Robust localization in wireless sensor networks via low-rank and sparse matrix decomposition. *International Conference on Future Networks and Distributed Systems*, 2017.

[153] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[154] W. Russel, D. Klein, and J. Hespanha. Optimal estimation on the graph cycle space. *IEEE Transactions on Signal Processing*, 59(6):2834 – 2846, 2011.

[155] V. Salari and I. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):87 – 91, 1990.

[156] J. Saunderson, P. A. Parrilo, and A. S. Willsky. Semidefinite descriptions of the convex hull of rotation matrices. *SIAM Journal on Optimization*, 25(3):1314 – 1343, 2015.

[157] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do I organize my holiday snaps?". In *Proceedings of the 7th European Conference on Computer Vision*, pages 414–431, 2002.

[158] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104 – 4113, 2016.

[159] P. Schroeder, A. Bartoli, P. Georgel, and N. Navab. Closed-form solutions to multiple-view homography estimation. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 650–657, Jan 2011.

[160] B. Servatius and W. Whiteley. Constraining plane configurations in computer-aided design: combinatorics of directions and lengths. *SIAM Journal on Discrete Mathematics*, 12(1):136 – 153, 1999.

[161] K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51 – 65, 2005.

[162] I. Shames, A. N. Bishop, and B. D. O. Anderson. Analysis of noisy bearing-only network localization. *IEEE Transactions on Automatic Control*, 58(1):247 – 252, 2013.

[163] G. C. Sharp, S. W. Lee, and D. K. Wehe. Multiview registration of 3D scenes by minimizing error between coordinate frames. In *Proceedings of the European Conference on Computer Vision*, pages 587–597, 2002.

[164] Y. Shen, Q. Huang, N. Srebro, and S. Sanghavi. Normalized spectral map synchronization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and

R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4925–4933. Curran Associates, Inc., 2016.

[165] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[166] S. Shih, Y. Chuang, and T. Yu. An efficient and accurate method for the relaxation of multiview registration error. *IEEE Transactions on Image Processing*, 17(6):968 – 981, 2008.

[167] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30(1):20 – 36, 2011.

[168] A. Singer and Y. Shkolnisky. Three-dimensional structure determination from common lines in cryo-EM by eigenvectors and semidefinite programming. *SIAM Journal on Imaging Sciences*, 4(2):543 – 572, 2011.

[169] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *SIGGRAPH: International Conference on Computer Graphics and Interactive Techniques*, pages 835–846, 2006.

[170] P. Stoica and K. C. Sharman. Maximum likelihood methods for direction of arrival estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7):1132 – 1143, 1990.

[171] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[172] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:525–653, 1999.

[173] M. Tao and X. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[174] R. E. Tarjan and U. Vishkin. An efficient parallel biconnectivity algorithm. *SIAM Journal on Computing*, 14(4):862 – 874, 1985.

[175] D. Thomas, Y. Matsushita, and A. Sugimoto. Robust simultaneous 3D registration via rank minimization. *Proceedings of the Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 33–40, 2012.

[176] R. Toldo, A. Beinat, and F. Crosilla. Global registration of multiple point clouds embedding the generalized procrustes analysis into an ICP framework. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2010.

[177] A. Torsello, E. Rodolà, and A. Albarelli. Multiview registration via graph diffusion of dual quaternions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2441 – 2448, 2011.

[178] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms*, pages 298–372. Springer-Verlag, 2000.

[179] R. Tron, L. Carlone, F. Dellaert, and K. Daniilidis. Rigid components identification and rigidity enforcement in bearing-only localization using the graph cycle basis. In *IEEE American Control Conference*, 2015.

[180] R. Tron and K. Danilidis. Statistical pose averaging with varying and non-isotropic covariances. In *Proceedings of the European Conference on Computer Vision*, 2014.

[181] R. Tron and R. Vidal. Distributed 3-D localization of camera sensor networks from 2-D image measurements. *IEEE Transactions on Automatic Control*, 59(12):3325–3340, 2014.

[182] R. Tron, X. Zhou, and K. Daniilidis. A survey on rotation optimization in structure from motion. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016.

[183] A. Tropp, N. Halko, and P. Martinsson. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions, 2010.

[184] C. Van Loan. The ubiquitous Kronecker product. *J. Comput. Appl. Math*, 123(1-2):85–100, 2000.

[185] V. S. Varadarajan. *Lie Groups, Lie Algebras, and Their Representations*, volume 102 of *Graduate Texts in Mathematics*. Springer, 1984.

[186] M. Wahlström. Half-integrality, LP-branching and FPT algorithms. pages 1762–1781, 2014.

[187] L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: a Journal of the IMA*, 2(2):145–193, 2013.

[188] Y.-X. Wang, C. M. Lee, L.-F. Cheong, and K.-C. Toh. Practical matrix completion and corruption recovery using proximal alternating robust subspace minimization. *International Journal of Computer Vision*, pages 1–30, 2014.

[189] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk. Sparcs: Recovering low-rank and sparse matrices from compressive measurements. In *Neural Information Processing Systems*, 2011.

[190] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

[191] W. Whiteley. Parallel redrawing of configurations in 3-space. Technical report, Champlain Regional College, St. Lambert, Quebec, Canada, 1986.

[192] W. Whiteley. Matroids from discrete geometry. In *Matroid Theory*, AMS Contemporary Mathematics, pages 171–313. American Mathematical Society, 1997.

[193] J. Williams and M. Bennamoun. Simultaneous registration of multiple corresponding point sets. *Computer Vision and Image Understanding*, 81(1):117–141, Jan. 2001.

[194] K. Wilson, D. Bindel, and N. Snavely. When is rotations averaging hard? In *Proceedings of the European Conference on Computer Vision*, pages 255 – 270, 2016.

[195] K. Wilson and N. Snavely. Robust global translations with 1DSfM. In *Proceedings of the European Conference on Computer Vision*, pages 61–75, 2014.

[196] C. Wu. Towards linear-time incremental structure from motion. In *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, 2013.

[197] J.-G. Yu, G.-S. Xia, A. Samal, and J. Tian. Globally consistent correspondence of multiple feature sets using proximal Gauss–Seidel relaxation. *Pattern Recognition*, 51:255 – 267, 2016.

[198] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1426 – 1433, 2010.

[199] C. Zeller and O. Faugeras. Camera self-calibration from video sequences: the Kruppa equations revisited. Research Report 2793, INRIA, Feb. 1996.

[200] Z. Zeng, T.-H. Chan, K. Jia, and D. Xu. Finding correspondence from multiple images via sparse and low-rank decomposition. In *Proceedings of the European Conference on Computer Vision*, pages 325 – 339, 2012.

[201] S. Zhao and D. Zelazo. Localizability and distributed protocols for bearing-based network localization in arbitrary dimensions. *Automatica*, 69:334 – 341, 2016.

[202] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust L_1-norm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, 2012.

[203] T. Zhou and D. Tao. GoDec: Randomized low-rank & sparse matrix decomposition in noisy case. In *Proceedings of the 28th International Conference on Machine Learning (ICML11)*, pages 33–40. ACM, June 2011.

[204] X. Zhou, C. Yang, H. Zhao, and W. Yu. Low-rank modeling and its applications in image analysis. *ACM Computing Surveys*, 47(2):36:1–36:33, 2014.

[205] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the International Conference on Computer Vision*, pages 4032 – 4040, 2015.

[206] Z. Zhou, X. Li, J. Wright, E. J. Candès, and Y. Ma. Stable principal component pursuit. In *IEEE International Symposium on Information Theory*, pages 1518–1522, 2010.